எளிய தமிழில்



Selenium

| து.நித்யா

எளிய தமிழில் Selenium

து.நித்யா

மென்பொருள் சோதனைத் துறையில், கட்டற்ற மென்பொருளான Selenium பெருமளவில் பயன்படுத்தப் படுகிறது. பெருகி வரும் இணைய தளங்களை தானியக்கமாக சோதிக்க இதைப் பயன்படுத்தலாம்.

இதை, இந்த நூல் எளிமையாக அறிமுகம் செய்கிறது.

தமிழில் கட்டற்ற மென்பொருட்கள் பற்றிய தகவல்களை "கணியம்" மின் மாத இதழ், 2012 முதல் வெளியிட்டு வருகிறது. இதில் வெளியான Selenium பற்றிய கட்டுரைகளை இணைத்து ஒரு முழு புத்தகமாக வெளியிடுவதில் பெரு மகிழ்ச்சி கொள்கிறோம்.

உங்கள் கருத்துகளையும், பிழை திருத்தங்களையும் editor@kaniyam.com க்கு மின்னஞ்சல் அனுப்பலாம்.

http://kaniyam.com/learn-selenium-in-tamil-ebook என்ற முகவரியில் இருந்து இந்த நூலை பதிவிறக்கம் செய்யலாம். உங்கள் கருத்துகளையும் இங்கே பகிரலாம்.

படித்து பயன் பெறவும், பிறருடன் பகிர்ந்து மகிழவும் வேண்டுகிறோம்.

கணியம் இதழை தொடர்ந்து வளர்க்கும் அனைத்து அன்பர்களுக்கும் எமது நன்றிகள்.

த.சீனிவாசன்
tshrinivasan@gmail.com

ஆசிரியர்
கணியம்
editor@kaniyam.com

எளிய தமிழில் Selenium

முதல் பதிப்பு ஆகஸ்டு 2016

ஆசிரியர் - து.நித்யா - nithyadurai87@gmail.com

பிழை திருத்தம்: த.சீனிவாசன் - tshrinivasan@gmail.com

வடிவமைப்பு: த.சீனிவாசன்

அட்டைப்படம் - லெனின் குருசாமி - gurulenin@gmail.com

நூல் மூலம் :

http://static.kaniyam.com/ebooks/learn-selenium-in-tamil/learn-selenium-in-tamil.odt

ஆசிரியர் உரை

கற்றலை நிறுத்தாதீர்கள்! உங்களின் வேலைக்கு நேரடியாகப் பயன்படும் விஷயத்தை மட்டும் கற்றுக்கொள்வதோடு நிறுத்தாமல், பல்வேறு வகையான விஷயங்களையும் தொடர்ந்து கற்றுக்கொண்டே இருங்கள். இதற்கு நீங்கள் பழகி விட்டால், அது உங்களின் தன்னம்பிக்கையை உயர்த்தும். உங்களின் சிந்தனையானது எப்போதும் புது உற்சாகத்துடனேயே காணப்படும். அந்த உற்சாகமும், தன்னம்பிக்கையும் உங்களின் நடை, உடை, பாவனை ஆகிய அனைத்திலும் வெளிப்படும். அதுவே பின்னாளில் உங்களை ஒரு மாமனிதராக மிகச் சிறந்த முறையில் செதுக்கும்.

வளர்ச்சியடைந்த நாடுகளில் உள்ள மக்கள், அவர்கள் என்ன வேலை செய்தாலும், அவர்களிடம் தன்னம்பிக்கையின்மையோ தாழ்வு மனப்பான்மையோ இருக்காது. ஏனென்றால் அவர்கள் புத்தகங்கள் படிப்பதை வாழ்க்கையின் ஒரு பகுதியாகவே எடுத்துக் கொள்கின்றனர். எனவே தான் அவர்களின் ஒவ்வொருவர் முகத்திலும் 'நான் தான் ராஜா' என்பது போன்ற பாவனை காணப்படும். ஆனால் நம் நாட்டிலோ கற்றல் என்பது வளமான வாழ்க்கைக்கு தேவைப்படும் வேலைக்கு உதவும் ஒரு கருவியாகவே கருதப்படுகிறது. பெரிய வேலை கிடைத்தவுடன் பலரும் புது விஷயத்தை கற்றுக் கொள்வதை நிறுத்தி விடுகின்றனர். எனவேதான் ஒருவரின் வேலையை பொறுத்து அவர்களின் தன்னம்பிக்கை நிலை இங்கு மாறுபடுவதைக் காணலாம்.

பல விஷயங்களை கற்றுத் தெரிந்துகொள்ளும்போதுதான் உங்களையே நீங்கள் ஒரு ராஜாவாக உணர்வீர்கள். இந்த உணர்வானது நாட்டில் நடக்கும் பல தவறுகளைத் தட்டிக் கேட்கும் தைரியத்தை உங்களுக்கு அளிக்கும். அந்த தைரியம் உண்மையிலேயே உங்களை ஒரு நாள் ராஜாவாக மாற்றும். உதாரணத்திற்கு கபாலி படத்தில் ரஜினி ஒரு பெரிய தலைவனாக நாட்டில் நடக்கும் பல தவறுகளை தட்டிக் கேட்பதுபோல் காணப்பட்டாலும், அவரின் அறிமுகக் காட்சியில், அவரும் ஒரு புத்தகத்தைப் படித்து ஏதோ ஒரு விஷயத்தைக் கற்றுக்கொண்டுதான் இருக்கிறார். நாட்டின் பெரிய பெரிய தலைவர்கள் எல்லாம் படிக்கும் பழக்கம் நிறைந்தவர்களே! ஆகவே கற்றலைக் கைவிடாதீர்கள்!

கற்றது கையளவு! கல்லாதது உலகளவு!

து. நித்யா

கிழக்கு தாம்பரம்,
சென்னை



27 ஆகஸ்டு 2016

மின்னஞ்சல்: *nithyadurai87@gmail.com*

வலைப்பதிவு: *http://nithyashrinivasan.wordpress.com*

இந்த நூலில் பயன்படுத்தப்பட்டுள்ள நிரல்கள் இங்கே உள்ளன. பதிவிறக்கிப் பயன்படுத்தலாம்.

https://github.com/nithyadurai87/code_selenium_python_ebook

தமிழில் காணொளி (வீடியோ) வடிவிலும் இந்த நூலின் பாடங்களை இங்கே காணலாம்.

பாகம் -1 *https://www.youtube.com/watch?v=qtyAWKIa8m4*

பாகம் -2 *https://www.youtube.com/watch?v=fuFYiubZlhQ*

பொருளடக்கம்

1 Automated Web application testing using Selenium

Selenium என்பது ஒரு browser automation tool ஆகும். இது உலகெங்கும் இருந்து பல்வேறு வல்லுனர்கள் இணைந்து உருவாக்கிய ஒரு கட்டற்ற மென்பொருள். இது 'Software Testing' துறையில் வேலை பார்ப்பவர்களுக்கு உதவும் ஒரு பயனுள்ள கருவி ஆகும்.

இதைக் கொண்டு Testing துறையில் உள்ளவர்கள் மிகவும் சுலபமாக அவர்களுடைய வேலைகளைச் செய்துவிட முடியும். இதைப் பற்றிக் கற்றுக்கொள்வதற்கு முன்னர் முதலில் நாம் browser, automation, tool எனும் ஒவ்வொரு வார்த்தையைப் பற்றியும் தெளிவாகப் புரிந்து கொள்ள வேண்டும். இதற்கு பின்வரும் கதையைப் படிக்கவும். அதன் பின்னரே நாம் selenium-ன் உதவியைக் கொண்டு எவ்வாறு பல்வேறு வேலைகளை விளையாட்டாகச் செய்வது எனக் கற்க முடியும்.

1.1 Automation - ஓர் அறிமுகம்

HTML / JavaScript கொண்டு ஒரு வலைத்தளமானது உருவாக்கப்படுகிறது. அவ்வாறு உருவாக்கப்பட்ட வலைத்தளமானது நேரடியாக பயனர்களைச் சென்றடைவதற்கு முன்னர் 'Testing' எனும் துறையின் கீழ் உள்ளவர்கள் அதனை பரிசோதிப்பார்கள். அவர்களுடைய வேலையே அந்த வலைத்தளத்திற்குள் ஒழுங்காக login செய்ய முடிகிறதா, உள்நுழைந்த பின்னர் பயனர்களுக்குத் தெரிய வேண்டிய அனைத்து இணைப்புகளும் ஒழுங்காகத் தெரிகிறதா, பயனர்களால் அந்த இணைப்புகளைப் பின்பற்றி அடுத்தடுத்த வேலைகளைச் செய்ய முடிகிறதா என்பதையெல்லாம் பரிசோதிப்பதே ஆகும். இதையெல்லாம் முதல் முறை செய்யும்போது பரிசோதிப்பவர்களுக்கு ஆர்வமாக இருக்கும். ஆனால் இதையே மீண்டும் மீண்டும் செய்ய வேண்டிய நிலை ஏற்படும் போது அவர்களுக்கும் ஆர்வம் குறைந்து விடும்.

அதாவது முதல் முறை வலைத்தளத்தை முழுவதுமாக பரிசோதித்து முடித்த பின்னர், வலைத்தளம் பயனர்களுக்கு வழங்கப்படும். பின்னர் பயனர்கள் அதனை அவர்களின் தொழிலுக்காக நேரடியாகப் பயன்படுத்தத் தொடங்கிவிடுவார்கள் [IT துறையில் இதனை Application (வலைத்தளம்) Production-க்குச் சென்று விட்டது என்று கூறுவோம்.] .

சிறிது காலப் பயன்பாட்டிற்குப் பின்னர், பயனர்கள் அவர்களின் தொழில் மாறுதலுக்கு ஏற்ப ஏதேனும் ஓர் இடத்தில் ஒரு புதிய படிவத்தை சேர்க்கவோ அல்லது மாற்றவோ வேண்டும் என்று கூறி மீண்டும் நம்மிடம் வருவார்கள். இந்நிலையில் Development துறையானது அவர்களின் விருப்பத்திற்கு ஏற்ப ஓர் புதிய படிவத்தை உருவாக்கி, வலைத்தளத்தின் ஓர் அங்கமாக இணைத்து விடும். இந்த வேலை முடிந்த பின்னர் வலைத்தளமானது மீண்டும் Testing

துறையின் கீழ் வரும். இப்போது *Testing* துறையில் உள்ளவர்கள் புதிதாக சேர்க்கப்பட்ட படிவத்தை மட்டும் பரிசோதிப்பத்துடன் நிறுத்திவிடாமல், மீண்டும் பழைய சோதனைகள் அனைத்தையும் பரிசோதிக்க வேண்டும். இதையே *'Regression Testing'* என்பார்கள்.

1.2 **Browser Automation** - ஓர் அறிமுகம்

Regression Testing- ன் படி வலைத்தளங்களில் ஒரு சிறிய மாறுதல் செய்யப்பட்டாலும் கூட மீண்டும் முதலிலிருந்து பரிசோதிக்க வேண்டும். அதாவது வலைத்தளத்திற்குள் ஒழுங்காக *login* செய்ய முடிகிறதா, உள்நுழைந்த பின்னர் பயனர்களுக்குத் தெரிய வேண்டிய அனைத்து இணைப்புகளும் ஒழுங்காகத் தெரிகிறதா, பயனர்களால் அந்த இணைப்புகளைப் பின்பற்றி அடுத்தடுத்த வேலைகளைச் செய்ய முடிகிறதா என்பதையெல்லாம் மீண்டும் மீண்டும் சோதிப்பதன் மூலம் சோதிப்பவர்களுடைய நேரமும் வீணாகிறது, அவர்களுக்கும் அலுப்பு தட்டுகிறது. எனவே இது போன்று திரும்ப திரும்ப செய்யப்படும் வேலைகளையெல்லாம் *browser*-யே வலைத்தளத்தில் செய்து நமக்கு வெறும் *result*-ஐ மட்டும் காட்டினால் எவ்வளவு நன்றாக இருக்கும் என்று ஒரு *Tester* நினைத்ததன் வெளிப்பாடே *Browser Automation* ஆகும். ஆம்! *Browser Automation* என்பது *Tester*-ஆகிய நாம் செய்யும் வேலைகளையெல்லாம் *browser*-யே செய்யவைத்து நமக்கு வெறும் *Pass/Fail* எனும் *result*-ஐ மட்டும் வெளிப்படுத்தும்.

1.3 **Selenium** கருவி - ஓர் அறிமுகம்

நாம் செய்யும் வேலைகளையெல்லாம் *browser*-ஐயே செய்ய வைப்பது எப்படி என்று ஒரு *Tester*-க்கு கற்றுக்கொடுக்கும் வகையில் *market*-ல் பல கருவிகள் உள்ளன. அவற்றில் *Rational Functional Tester, Test Complete, Unified Functional Tester, Selenium* ஆகியவை மிகவும் பிரபலமானவை ஆகும். இவற்றில் *Selenium* தவிர மற்றவையெல்லாம் நாம் காசு கொடுத்து வாங்கிய பின்னரே, அதனை பயன்படுத்த முடியும். *Selenium* மட்டுமே இலவசமாகக் கிடைக்கக்கூடிய ஒன்று. எனவே *Selenium* மூலம் *automation*-ஐ இந்தப் புத்தகத்தில் நாம் பயிற்சி செய்து பார்க்கலாம். *Automation*- க்காக *selenium*-ல் *4* வகைக் கருவிகள் உள்ளன. அவை பின்வருமாறு.

1. *Selenium IDE*
2. *Selenium WebDriver*
3. *Selenium Standalone Server*
4. *Selenium Grid*

1.3.1 **Selenium IDE**

'Selenium IDE' என்பது சொல்வதைச் சொல்லும் கிளிப்பிள்ளை. இது *Firefox browser*-ன் *plug-in* ஆக வரும். இந்த *IDE*-ஐத் திறந்து வைத்துக்கொண்டு *firefox browser*-ல் என்ன செய்தாலும், அதை அப்படியே *IDE* பதிவு செய்து கொள்ளும்.

இந்தப் பதிவினை நாம் சேமித்துவிட்டு, மீண்டும் நமக்கு எப்போது தேவையோ, அப்போது எடுத்து *Run* செய்வதன் மூலம், நாம் பதிவின்போது என்னென்னவெல்லம் *browser*-ல் செய்தோமோ, அவை அப்படியே தானாகவே *browser*-ல் செய்யப்படும். ஆனால் இது ஒரு முழுமையான *'Automation Framework'* உருவாக்குவதற்கு உதவாது.

1.3.2 **Selenium WebDriver**

'Selenium WebDriver' என்பது ஒரு முழுமையான *'Automation Framework'* உருவாக்குவதற்குப் பயன்படுகிறது. அதாவது வலைத்தளப் பக்கத்தின் *content*-ஐ ஒப்பிடுவது, அதனடிப்படையில் ஒருசில செயல்களைச் செய்வது, இறுதி மதிப்புகளை எங்காவது பதிவு செய்வது போன்ற அனைத்தையும் உள்ளடக்கியதே *'Automation Framework'* ஆகும். இவையெல்லாம் செய்வதற்கு *IDE* உதவாது. *WebDriver*-தான் இதற்கு ஒரு சிறந்த கருவி ஆகும். எனவே பரவலாகப் பயன்படுத்தப்படும் *Automation Projects*-ல் இந்த *'Selenium Webdriver'*-தான் முழுப்பங்கு வகிக்கும்.

1.3.3 **Selenium Standalone Server**

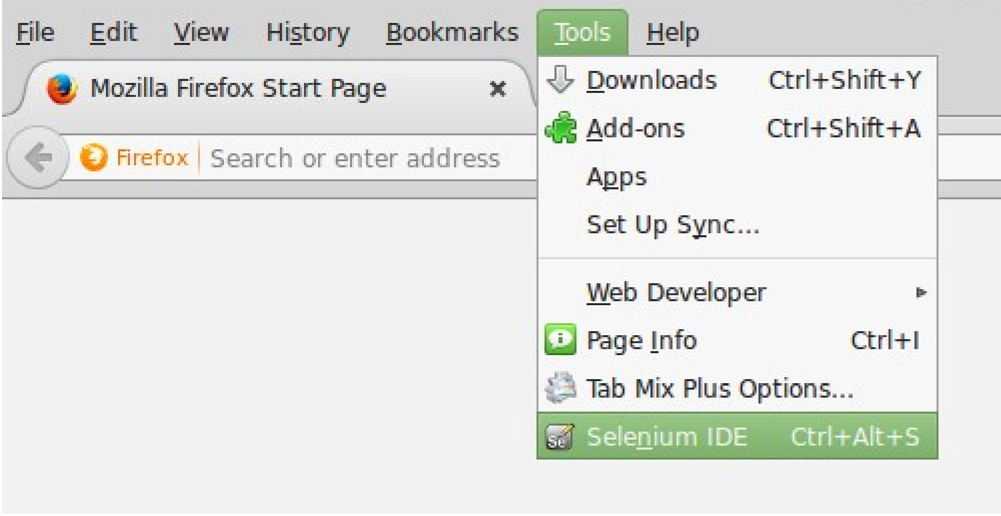
'Selenium Standalone server' என்பது நமது கணினியில் மட்டுமல்லாமல் தொலைதூரக் கணினிகளுடன் தொடர்பு கொண்டு, அதிலும் சோதனைகளை நிகழ்த்தப் பயன்படும் ஒரு கருவி ஆகும். அதாவது நாம் லினக்ஸ் கணினியில் உள்ள *firefox*-ல் சோதனைகளை நிகழ்த்திக் கொண்டிருக்கும்போது, அதே சோதனைகளை *IE*- ல் நிகழ்த்த விரும்பினால், *IE browser* லினக்ஸ் கணினியில் இருக்காது. எனவே இந்த *'Selenium Standalone server'* மூலம் மற்றொரு *windows* கணினியுடன் தொடர்பு கொண்டு, நம்மால் சோதனைகளை *IE*-ல் நிகழ்த்த முடியும். இதுவே இதன் பயன்பாடு ஆகும்.

1.3.4 **Selenium Grid**

'Selenium Grid' ஆனது ஒரே நேரத்தில் பல்வேறு கணினிகளுடன் தொடர்பு கொண்டு, நமது சோதனைகளை பல்வேறு வகையான *browser*-களில் நிகழ்த்த உதவும். வேறுபட்ட இடங்களில் இருக்கும், வெவ்வேறு வகையான இயங்குதளங்களைக் கொண்ட கணினிகளில் ஒரே சமயத்தில், குறைந்த நேரத்தில் சோதனைகளை நிகழ்த்தி முடிப்பதே இதன் சிறப்பு ஆகும்.

2. Selenium IDE

http://www.seleniumhq.org/download/ எனும் முகவரியில் சென்று *selenium IDE Version 2.9.0* ஐ *install* செய்யவும். இது *install* செய்யப்பட்டு விட்டதா என்பதை பரிசோதிக்க *firefox browser*-ஐ ஒருமுறை *close* செய்துவிட்டு மீண்டும் திறக்கவும். பின்னர் *Tools*-ன் *submenu*-ஆக *Selenium IDE* தெரிகிறதெனில் அது *install* செய்யப்பட்டுவிட்டது என்று அர்த்தம். இது *firefox browser*-ன் *plugin* ஆகவும் வரும்.



Wordpress-க்குள் சென்று ஒரு புதிய *blog*-ஐ உருவாக்கி வெளியிடும் விதத்தை எவ்வாறு *IDE* - மூலம் தானாக இயங்க வைப்பது என்று இப்போது பார்க்கலாம்.

அதற்கு முன்னர் ஒரு புதிய *blog*-ஐ உருவாக்குவதற்கான படிகள் பின்வருமாறு.

2.1 Manual Process

1. புதிதாக ஒரு *firefox browser*-ஐ திறக்கவும்.

2. *https://valaipathivu.wordpress.com/wp-admin* எனும் முகவரியை அடித்து *enter* செய்யவும்.

3. பின்வரும் தகவல்களைக் கொடுத்து *Log In*-ஐ சொடுக்கவும்.
Username: valaipathivu
Password: Kadavuchol

4. உள் நுழைந்தவுடன் வலைத்தளத்தின் இடப்பக்கத்தில் உள்ள Posts எனும் இணைப்பின்மீது சொடுக்கவும்.

5. பின்னர் அதன் submenu-வாகத் தெரிபவற்றில், Add New இணைப்பின்மீது சொடுக்கவும்.

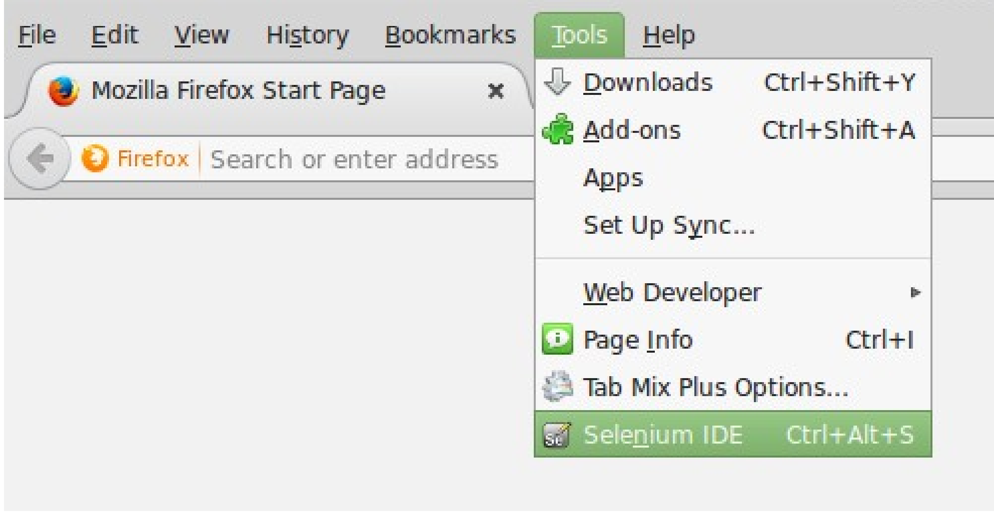
6. இப்போது தெரியும் 'Add New Post' படிவத்தில் 'Enter title here' எனத் தெரியும் இடத்தில், 'Tamil Kavithaikal' எனக் கொடுக்கவும்.

7. பின்னர் வலப்பக்கத்தில் உள்ள Publish எனும் பொத்தானின் மீது சொடுக்கவும்.

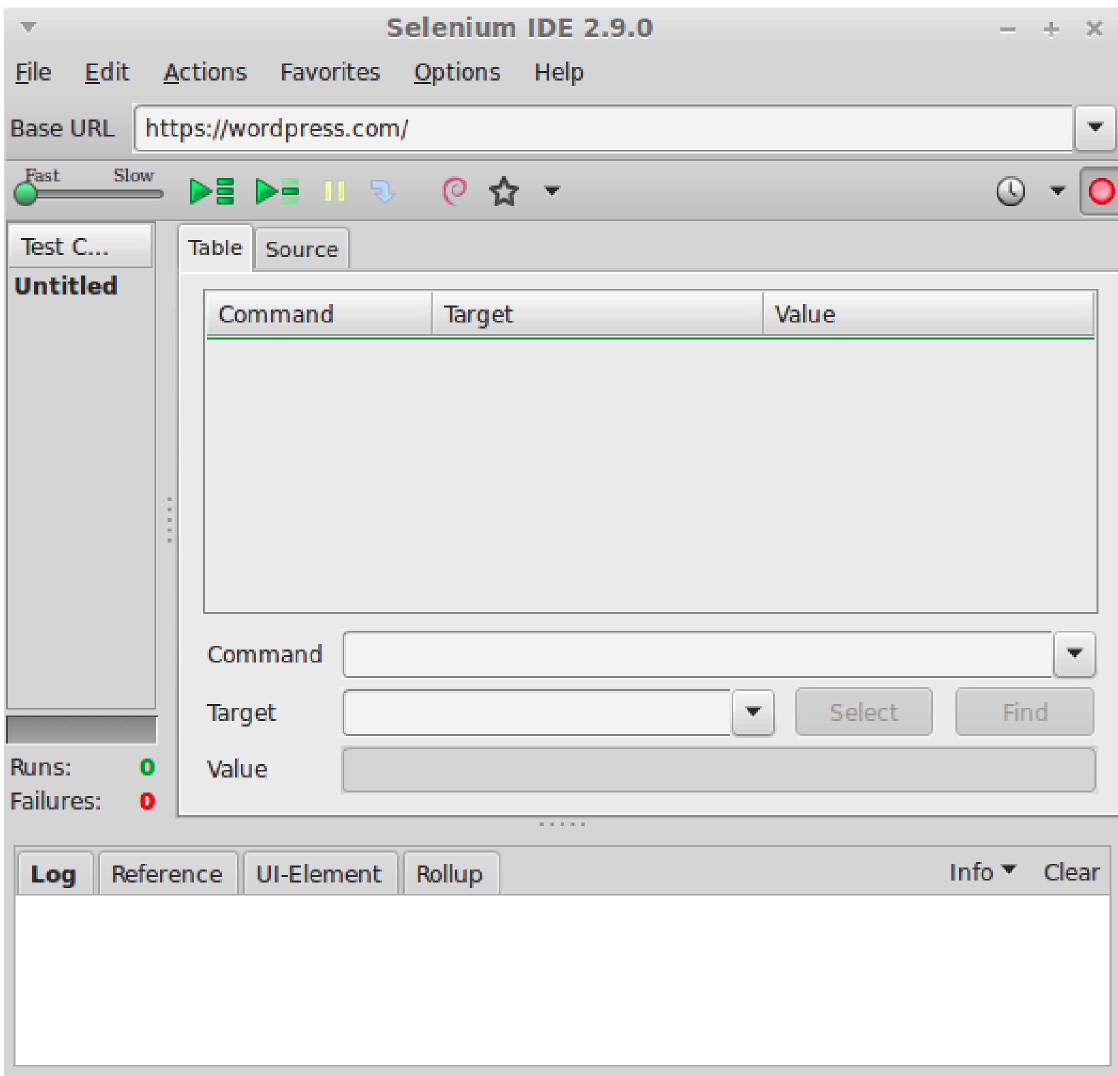
இப்போது selenium IDE-ஆல் automate செய்வதில் உள்ள படிகளை பின்வருமாறு காணலாம்.

2.2 **Automation Process**

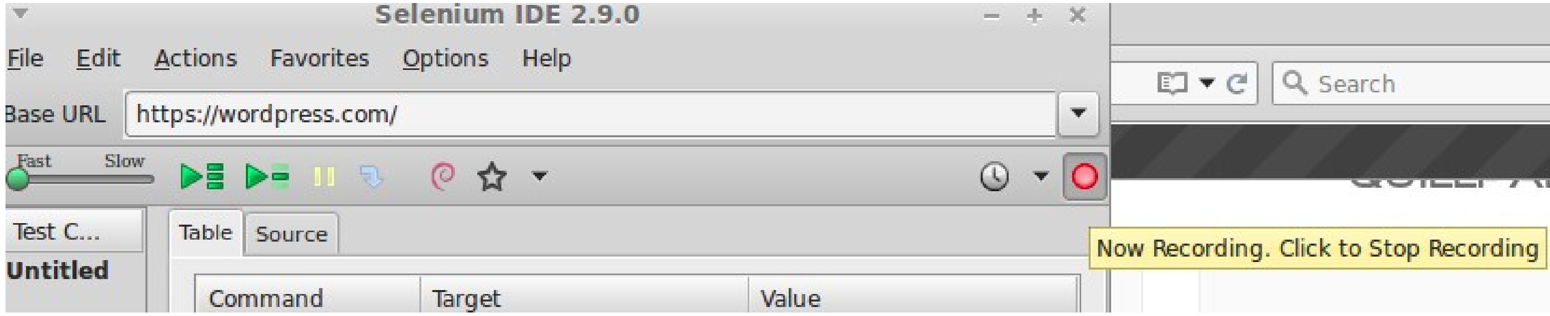
1. புதிதாக ஒரு firefox browser-ஐ திறக்கவும். அதன் menubar-ல் Tools -> Selenium IDE என்பதன் மீது சொடுக்கவும்.



2. Selenium IDE-ஆனது திறக்கப்பட்டு பின்வருமாறு காணப்படும்.



*3. IDE-*ஆனது திறக்கப்படும்போதே *recording* நிலையிலேயே இருக்கும். எனவே தனியாக ஒருமுறை சென்று சிகப்பு நிற வட்ட வடிவில் இருக்கும் பொத்தானை சொடுக்கத் தேவையில்லை. இதனை நீங்கள் உறுதி செய்ய விரும்பினால் *record-*ன் மீது சென்று *cursor-*ஐ வைக்கவும். *"Now Recording. Click to Stop Recording"* என்று காட்டுகிறது. எனவே இன்னொரு முறை சொடுக்கினால் *recording* நின்று விடும் என்பதை நினைவில் வைத்துக் கொள்ளவும்.



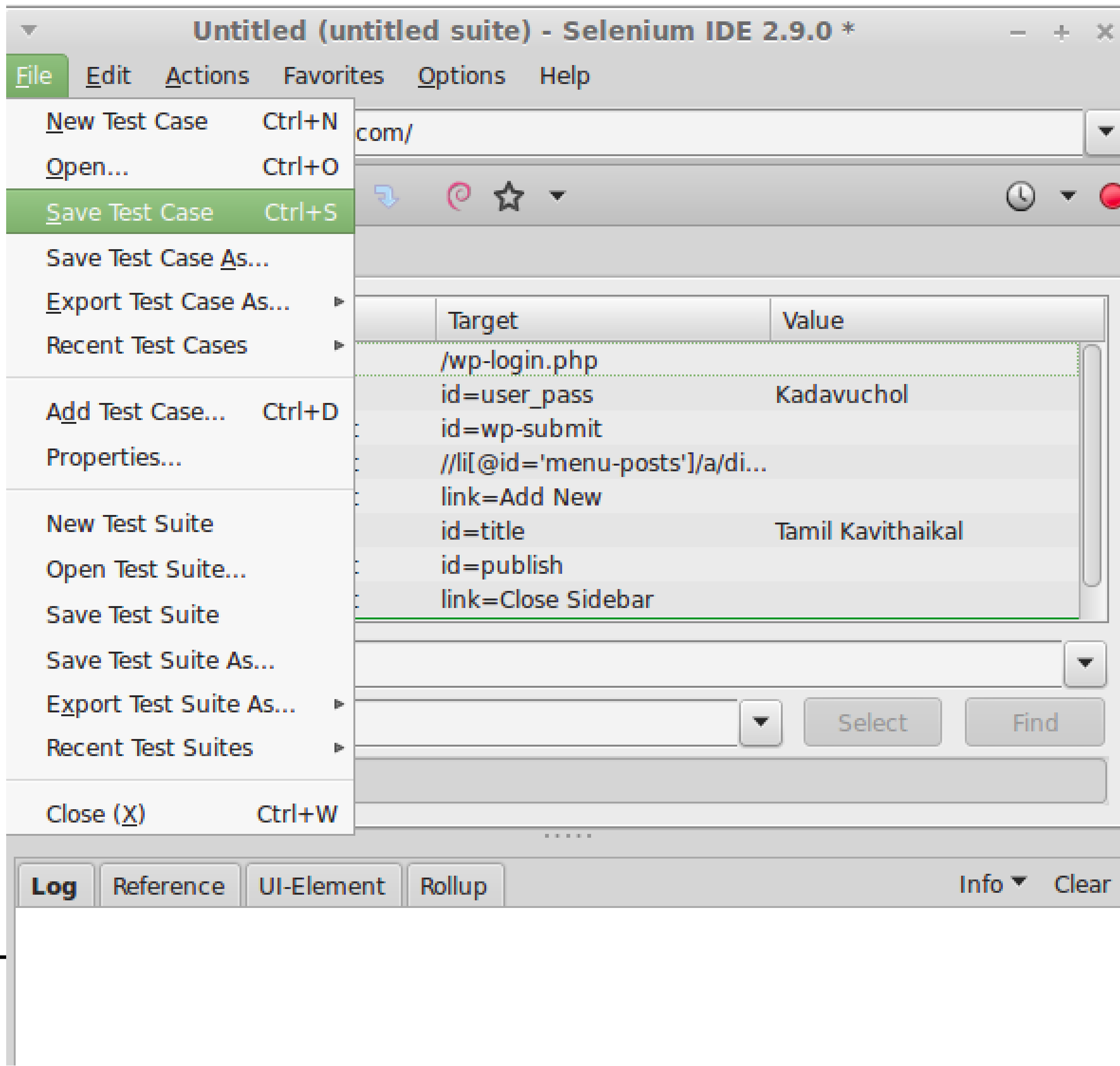
4. இந்நிலையில் *browser-*ல் சென்று *'Manual Process'* உள்ள *2-*வது படி முதல் *7-*வது படிவரை ஒரு முறை செய்து முடித்து பின்னர் மீண்டும் *IDE-*ல் வந்து அந்த சிகப்பு நிற பொத்தானை சொடுக்கவும்.

5. இதுவரை நீங்கள் *browser-*ல் செய்த அனைத்து வேலைகளும் *Command, Target, Value* வடிவில் *IDE-*ஆல் *record* செய்யப்பட்டுள்ளது.

Table | Source

Command	Target	Value
open	/wp-login.php?redirect_to...	
type	id=user_login	valaipathivu
type	id=user_pass	Kadavuchol
clickAndWait	id=wp-submit	
clickAndWait	//li[@id='menu-posts']/a/di...	
clickAndWait	link=Add New	
type	id=title	Tamil Kavithaikal
clickAndWait	id=publish	

6. இதனை அப்படியே ஒரு *test case*-ஆக சேமிக்கவும்.



7. இங்கு நான் *sample* எனும் பெயரில் *test case*-ஐ சேமித்துள்ளேன். எனவே அந்த பெயர் *IDE*-ன் மேல்புறத்திலும் பக்கவாட்டிலும் தெரிவத்தை கவனிக்கவும்.

sample (untitled suite) - Selenium IDE 2.9.1

File Edit Actions Favorites Options Help

Base URL https://valaipathivu.wordpress.com/

Fast Slow

Test C...
sample

Table Source

Command	Target	Value
open	/wp-login.php?redirect_to...	
type	id=user_login	valaipathivu
type	id=user_pass	Kadavuchol
clickAndWait	id=wp-submit	
clickAndWait	//li[@id='menu-posts']/a/di...	
clickAndWait	link=Add New	
type	id=title	Tamil Kavithaikal
clickAndWait	id=publish	

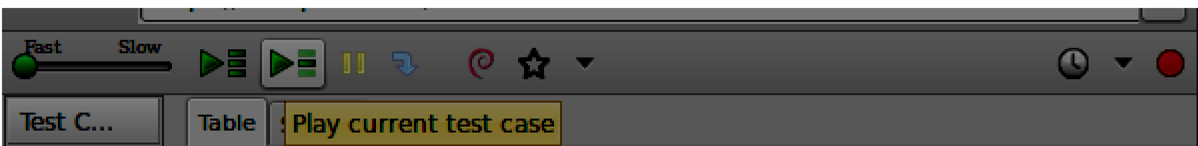
Command clickAndWait
Target id=publish Select Find
Value

Runs: 1
Failures: 0

Log Reference UI-Element Rollup Info Clear

[info] Playing test case sample
[info] Executing: |open | /wp-login.php?redirect_to=https%3A%2F%2Fvalaipathivu.wordpress.com%2Fwp-admin%2F&reauth=1 | |
[info] Executing: |type | id=user_login | valaipathivu |
[info] Executing: |type | id=user_pass | Kadavuchol |

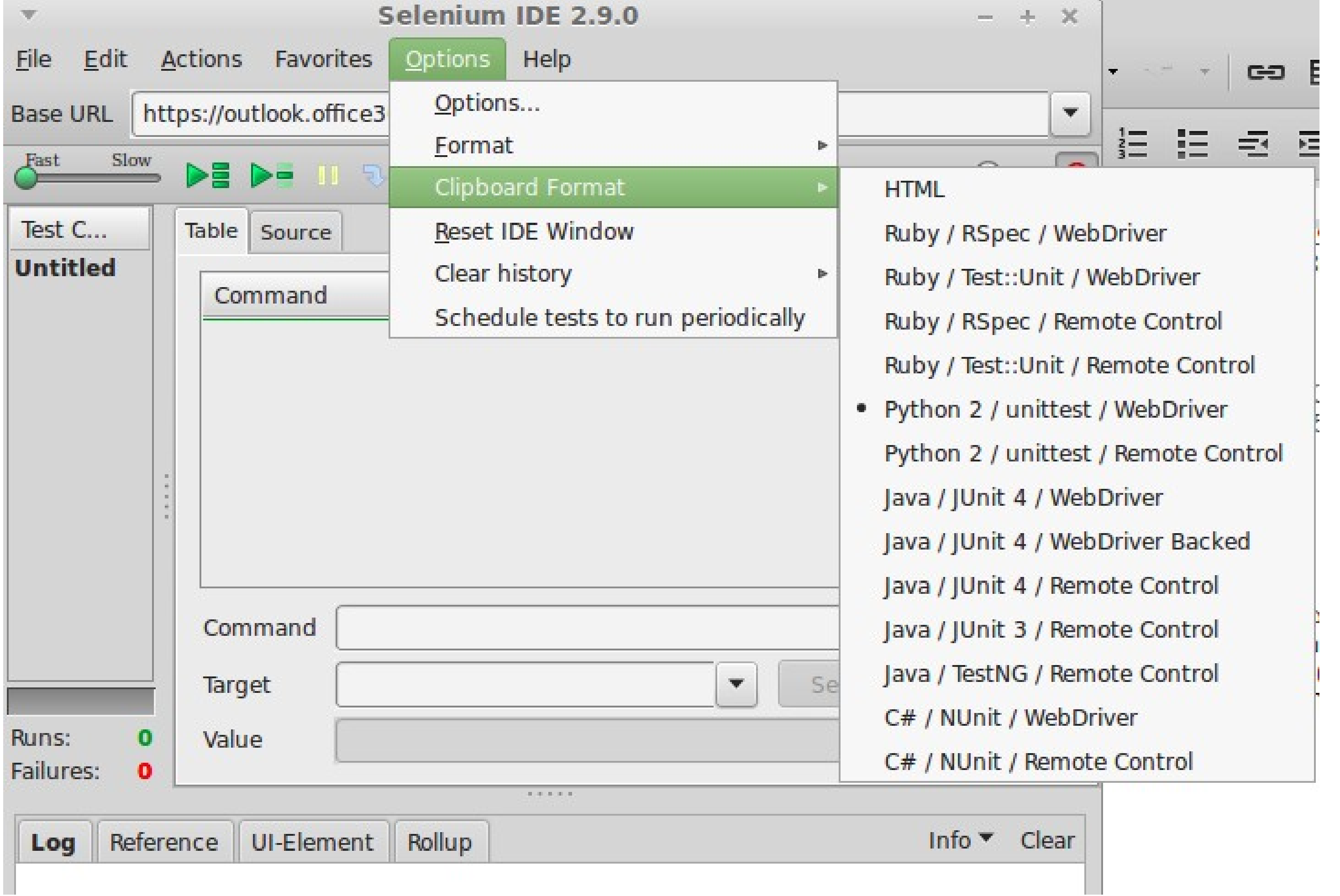
8. இப்போது படுத்த முக்கோண வடிவில் இருக்கும் *'Play current test case* எனும் பொத்தானைஅழுக்கினால், நாம் செய்த விஷயங்கள் அனைத்தையும் *browser* தானாகவே செய்து முடிப்பதை *firefox*-ல் காணலாம்.



9. கடைசியாக execution முடிந்தவுடன், அதற்கான log பின்வருமாறு காணப்படும்.

```
[info] Executing: |open | /wp-login.php?redirect_to=https%3A%2F
%2Fvalaipathivu.wordpress.com%2Fwp-admin%2F&reauth=1 | |
[info] Executing: |type | id=user_login | valaipathivu |
[info] Executing: |type | id=user_pass | Kadavuchol |
[info] Executing: |clickAndWait | id=wp-submit | |
[info] Executing: |clickAndWait | //li[@id='menu-posts']/a/div[3] | |
[info] Executing: |clickAndWait | link=Add New | |
[info] Executing: |type | id=title | Tamil Kavithaikal |
[info] Executing: |clickAndWait | id=publish | |
[info] Test case passed
```

2.3 IDE-ன் நிறை குறைகள்



இப்போது *IDE* மூலமே ஒரு விஷயத்தை சுலபமாகச் செய்யும்போது, எதற்காக ஒரு மொழியைக் கற்றுக்கொண்டு நிரல் எழுதி *webdriver* மூலம் செய்ய வேண்டும் என்று யோசிக்கிறீர்களா? உதாரணத்துக்கு *wordpress*-ல் உள்நுழைந்தவுடன் *'Login is successful'* , புதிய பதிவு வெளியானவுடன் *'New post is published'* என்பது போன்ற வெளிப்பாடுகளை பயனர்களுக்குத் தெரிவிக்க *IDE*-ஆல் முடியாது. அதற்கு நாம் தான் *selenium* ஆதரிக்கும் மொழிகளில் ஒன்றில் நிரல் எழுத வேண்டும். இது *html, ruby, python, java, C#* எனும் *5* மொழிகளை ஆதரிக்கும். இந்த மொழிகளின் பட்டியலை *IDE*-ல் பின்வருமாறு காணலாம்.

நிரல் எழுதுவதற்கு *IDE*-ஆனது நமக்கு ஒருவிதத்தில் உதவி புரியும். அதாவது, நாம் பதிவு செய்த *recording*-க்கு *selenium* ஆதரிக்கும் மொழிகளில் *code*-ஐ உருவாக்கிக் கொடுத்துவிடும். *File -> Export Test Suite As ->* என்பதன் மூலம் நமக்குத் தேவைப்படும் மொழிகளில் நாம் நம் நிரல்களின் கோப்பினை பெற்றுக்கொள்ள முடியும். எனவே இந்தப் பண்பினைப் பயன்படுத்தி நான் *automate* செய்ததற்கான *python* கோப்பினை *wordpress.py* என சேமித்துக்

கொள்கிறேன்.

இப்போது இதனை *run* செய்வதற்கு முன்னர் *selenium webdriver*-ஐ *install* செய்து கொள்ளவும். அப்போதுதான் இதுபோன்ற நிரல்களின் கோப்பு இயக்கப்படும். *Webdriver*- ஐ *install* செய்தவுடன் *terminal*-ல் சென்று *python wordpress.py* எனக் கொடுத்து ஒரு சாதாரண *python program*-ஐ எவ்வாறு *run* செய்வோமோ அவ்வாறு *run* செய்கிறேன்.

```
nithya@nithya-laptop: ~/Desktop/selenium
File  Edit  View  Search  Terminal  Help
nithya@nithya-laptop:~/Desktop/selenium$ ls
All   kaniyam                     Selenium.odt   snapshot5.png  wordpress.py
a.py  kaniyampython.py            snapshot2.png  snapshot6.png
ex1   LearningSelenium-sample.pdf snapshot3.png  snapshot7.png
ex2   sample                      snapshot4.png  Wordpress
nithya@nithya-laptop:~/Desktop/selenium$ python wordpress.py
```

இப்போது *browser*- ல் தானாக அடுத்தடுத்த செயல்கள் நடந்து கொண்டிருக்கும்போது *'Add New'* எனும் இணைப்பினை காணவில்லை எனும் ஒரு *error* வந்து எனது *script, fail* ஆகிறது. இது பின்வருமாறு.

nithya@nithya-laptop: ~/Desktop/selenium

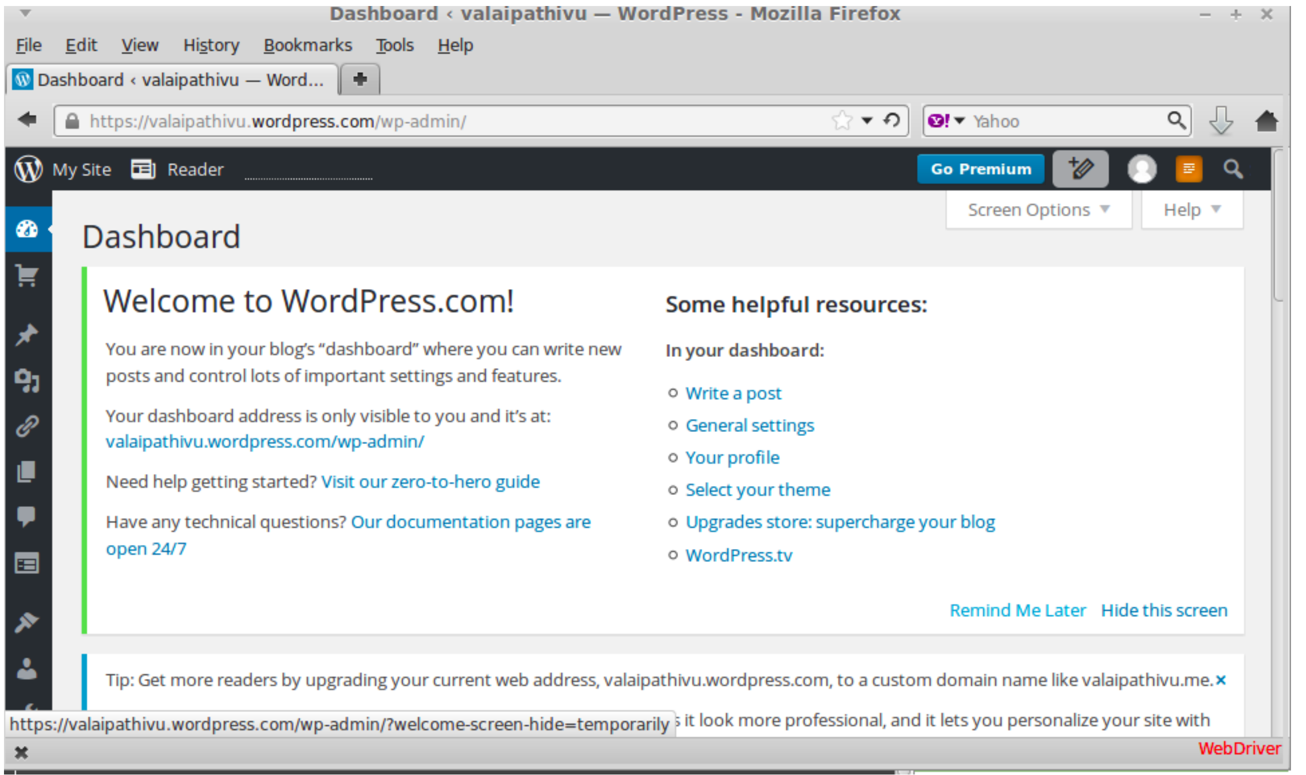
File Edit View Search Terminal Help

```
nithya@nithya-laptop:~/Desktop/selenium$ python wordpress.py
E
======================================================================
ERROR: test_wordpresspython (__main__.Wordpresspython)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "wordpress.py", line 26, in test_wordpresspython
    driver.find_element_by_link_text("Add New").click()
  File "/usr/local/lib/python2.7/dist-packages/selenium/webdriver/remote/webdriv
er.py", line 274, in find_element_by_link_text
    return self.find_element(by=By.LINK_TEXT, value=link_text)
  File "/usr/local/lib/python2.7/dist-packages/selenium/webdriver/remote/webdriv
er.py", line 692, in find_element
    {'using': by, 'value': value})['value']
  File "/usr/local/lib/python2.7/dist-packages/selenium/webdriver/remote/webdriv
er.py", line 193, in execute
    self.error_handler.check_response(response)
  File "/usr/local/lib/python2.7/dist-packages/selenium/webdriver/remote/errorha
ndler.py", line 181, in check_response
    raise exception_class(message, screen, stacktrace)
NoSuchElementException: Message: Unable to locate element: {"method":"link text"
,"selector":"Add New"}
Stacktrace:
    at FirefoxDriver.prototype.findElementInternal_ (file:///tmp/tmpCUL5Tt/exten
sions/fxdriver@googlecode.com/components/driver-component.js:10667)
    at fxdriver.Timer.prototype.setTimeout/<.notify (file:///tmp/tmpCUL5Tt/exten
sions/fxdriver@googlecode.com/components/driver-component.js:623)

----------------------------------------------------------------------
Ran 1 test in 38.435s

FAILED (errors=1)
nithya@nithya-laptop:~/Desktop/selenium$
```

இதற்கான காரணம் என்னவெனில், *webdriver* மூலம் *run* செய்யும்போது ஒரு புதிய *browser* பின்வரும் அளவில் உருவாகி, அதில் செயல்கள் எல்லாம் நடைபெற்றுக்கொண்டிருக்கும்.



இந்த அளவில் இருக்கும் *browser-ல் 'Add New'* எனும் இணைப்பு தெரிவதற்கு வாய்ப்பில்லை. எனவே தான் *Add New* இணைப்பைக் காணவில்லை எனும் தவறு நடக்கிறது. எனவே இவ்வாறு இருக்கும் நிரலை மாற்றுவதற்கு பதிலாக நாமே நிரலை எழுதி விடலாம். ஆதலால் இதுபோன்ற விஷயங்களைச் சும்மா தெரிந்து வைத்துக் கொள்ளுங்கள். ஆனால் இதையே முழுதாக நம்பி இருக்காதீர்கள்.

3. **Selenium Webdriver**

உங்களுடையது linux கணிணியாக இருந்தால் terminal-ல் சென்று sudo pip install selenium என்று கொடுக்கவும். இது selenium webdriver-ஐ install செய்துவிடும். அப்படியே python-ஐயும் install செய்து கொள்ளவும்.

Wordpress-க்குள் சென்று ஒரு புதிய blog-ஐ உருவாக்கி வெளியிடும் விதத்தை Webdriver - மூலம் தானாக இயங்க வைப்பதற்கான python code இங்கு கொடுக்கப்பட்டுள்ளது.

```
#File : wordpress.py
from selenium import webdriver

a = webdriver.Firefox()
a.get("https://valaipathivu.wordpress.com/wp-admin")
a.maximize_window()

a.find_element_by_xpath("//input[@id='user_login']").send_keys("valaipathivu")
a.find_element_by_xpath("//input[@id='user_pass']").send_keys("Kadavuchol")
a.find_element_by_xpath("//input[@id='wp-submit']").click()

print "Login is successful"

a.find_element_by_link_text("Posts").click()
a.find_element_by_link_text("Add New").click()
a.find_element_by_xpath("//input[@id='title']").send_keys("Tamil Kavithaikal")
a.find_element_by_xpath("//input[@id='publish']").click()

print "New post is Published"
```

இப்போது மேற்கண்ட python code-ன் ஒவ்வொரு வரிக்கான விளக்கத்தையும் பின்வருமாறு காணலாம்.

1. selenium-ல் உள்ள webdriver எனும் கருவி நமது வலைத்தளப் பக்கங்களின் html மொழியுடன் தொடர்பு கொண்டு, ஒரு பயனர் என்னென்ன செய்ய வேண்டுமோ அதையெல்லாம் இந்தக் கருவியே செய்து விடும். எனவே இதனை முதலில் நமது program-ல் ஏற்றுமதி செய்துவிட வேண்டும். இதையே பின்வரும் வரி விளக்குகிறது.
from selenium import webdriver

2. அடுத்து webdriver எந்த browser-ஐ பயன்படுத்த வேண்டும் என்பது பின்வருமாறு குறிப்பிடப் பட்டுள்ளது. மேலும் இந்த விஷயம் நாம்

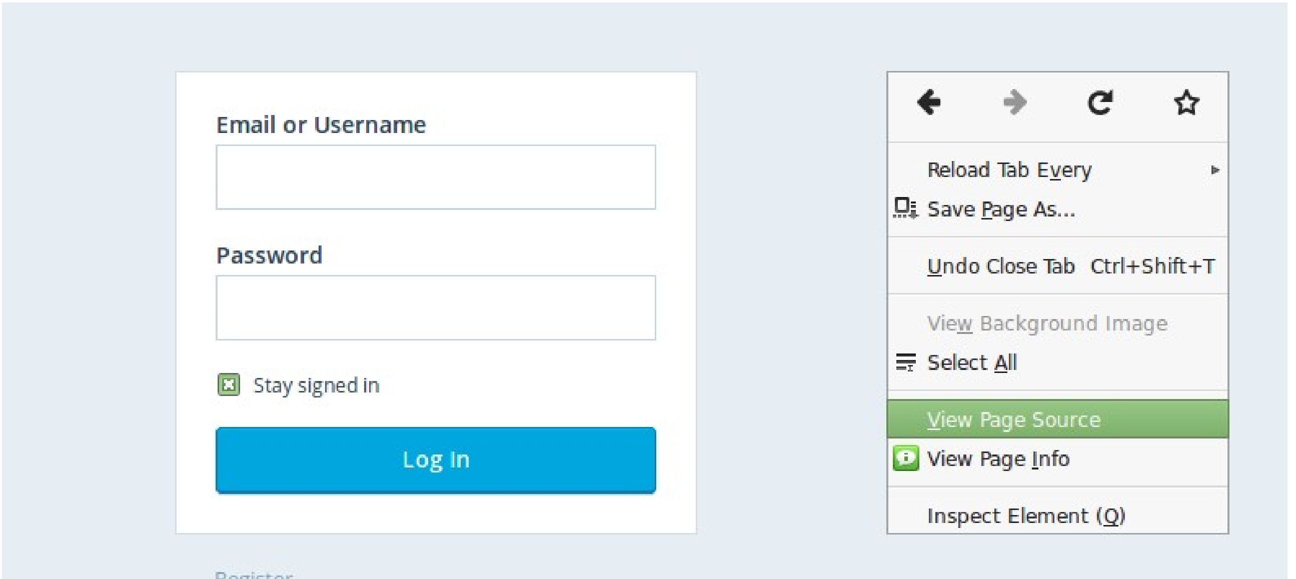
எழுதப்போகும் ஒவ்வொரு வரியின் தொடக்கத்திலும் குறிப்பிடப்பட வேண்டும் என்பதால் இந்த விஷயத்தை *a* எனும் *variable*-க்கு *assign* செய்துள்ளோம்.
a = webdriver.Firefox()

3. இப்போது *get()* எனும் *function* மூலம் எந்த முகவரியைப் பயன்படுத்த வேண்டும் எனக் குறிப்பிட்டுள்ளோம்.
a.get("https://valaipathivu.wordpress.com/wp-admin")

4. *maximize_window()* எனும் *function* நமது *browser*-ஐ பெரிதுபடுத்த உதவும்.

5. அடுத்ததாக நாம் *username* மற்றும் *password*-ஐ *wordpress*-க்குள் செலுத்த வேண்டும். *Webdriver*-ஆனது வலைத்தளப் பக்கத்தில் உள்ள *textbox*-ன் *id*-யை வைத்து அந்த இடத்திற்குச் செல்ல *find_element_by_xpath()* எனும் *function*-ஐயும், அவ்வாறு கண்டுபிடிக்கப்பட்ட *textbox*-க்குள் மதிப்புகளைச் செலுத்த *send_keys()* எனும் *function*-ஐயும் பயன்படுத்துகிறது.

இப்போது அந்த இரண்டு *textbox*-ன் *html id*-ஐக் கண்டுபிடிக்க, அந்தப் பக்கத்தின் மீது *'Rightclick'* செய்து *'View Page Source '* என்பதின் மீது சொடுக்கவும்.



இது பின்வருமாறு ஒரு *html file*-ஐக் காட்டும்.

```
Source of: https://wordpress.com/wp-login.php - Mozilla Firefox
File  Edit  View  Help

1  <!DOCTYPE html>
2      <!--[if IE 8]>
3          <html xmlns="http://www.w3.org/1999/xhtml" class="ie8" lang="en">
4      <![endif]-->
5      <!--[if !(IE 8) ]><!-->
6          <html xmlns="http://www.w3.org/1999/xhtml" lang="en">
7      <!--<![endif]-->
8      <head>
9      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
10     <title>WordPress.com &rsaquo; Log In</title>
11     <link rel='stylesheet' id='all-css-0' href='https://s0.wp.com/wp-includes/css/buttons.min.css?m=1408088319g' type='text/css' media='all' />
12 <link rel='stylesheet' id='open-sans-css'  href='https://fonts.googleapis.com/css?family=Open+Sans%3A300italic%2C400italic%2C600italic%2C300%2C400%2C600&#038;subset=latin%2Cla
13 <link rel='stylesheet' id='all-css-2' href='https://s2.wp.com/_static/??/wp-includes/css/dashicons.min.css,/wp-admin/css/login.min.css?m=1438861833j' type='text/css' media='al
14 <link rel='stylesheet' href='https://s0.wp.com/wp-admin/css/wpcom.css?m=1439241376g&#038;version=4.3-alpha-32859' type='text/css' />
15 <script type="text/javascript">
16 if ( window.top !== window.self ) {
17     if ( window.top.location.hostname !== window.self.location.hostname || window.top.location.protocol !== window.top.location.protocol ) {
18         window.top.location.href = window.self.location.href;
19     }
20 }
21 </script>
22 <script type='text/javascript' src='https://s2.wp.com/_static/??-eJzTLy/QzcxLzilNSS3WzwKiwtLUokoopZdVrKOPT4FubmZ6UWJJql5uZh5QsX2uraGJsbGpoZGJkUkWACVBIKM='></script>
23 <link rel="canonical" href="https://wordpress.com/wp-login.php">
24     <style type="text/css">
25         body, form .input {
26             font-family: "Open Sans", "Helvetica Neue", "Arial", sans;
27         }
28
29         body.login.recover-password,
30         body.login.recover-password-success {
31             height: auto;
32             min-height: 100%;
33         }
34
35
36         #recovery {
37             padding-bottom: 24px;
38         }
39
40         .recover-password #login_error {
41             width: auto;
42         }
43
44         #login_error ul {
45             list-style: square outside none;

Menu | [Inbox - nit... | WordPress.... | Pharmacy | Selenium.o... | Sakthikann... | Pharmacy.o... | Source of: ... | 19 °C  Tue Aug 11, 11:44 AM
```

இதில் சென்று நமக்கு வேண்டிய இரண்டு *textbox*-ன் *id*-ஐக் கண்டுபிடிப்பது சற்று கடினமான விஷயம் தான். எனவே இதற்கான மாற்றுவழி என்னவெனில், *browser*-ல் சென்று *Tools -> Web Developer -> Web Console* என்பதை சொடுக்கவும்.

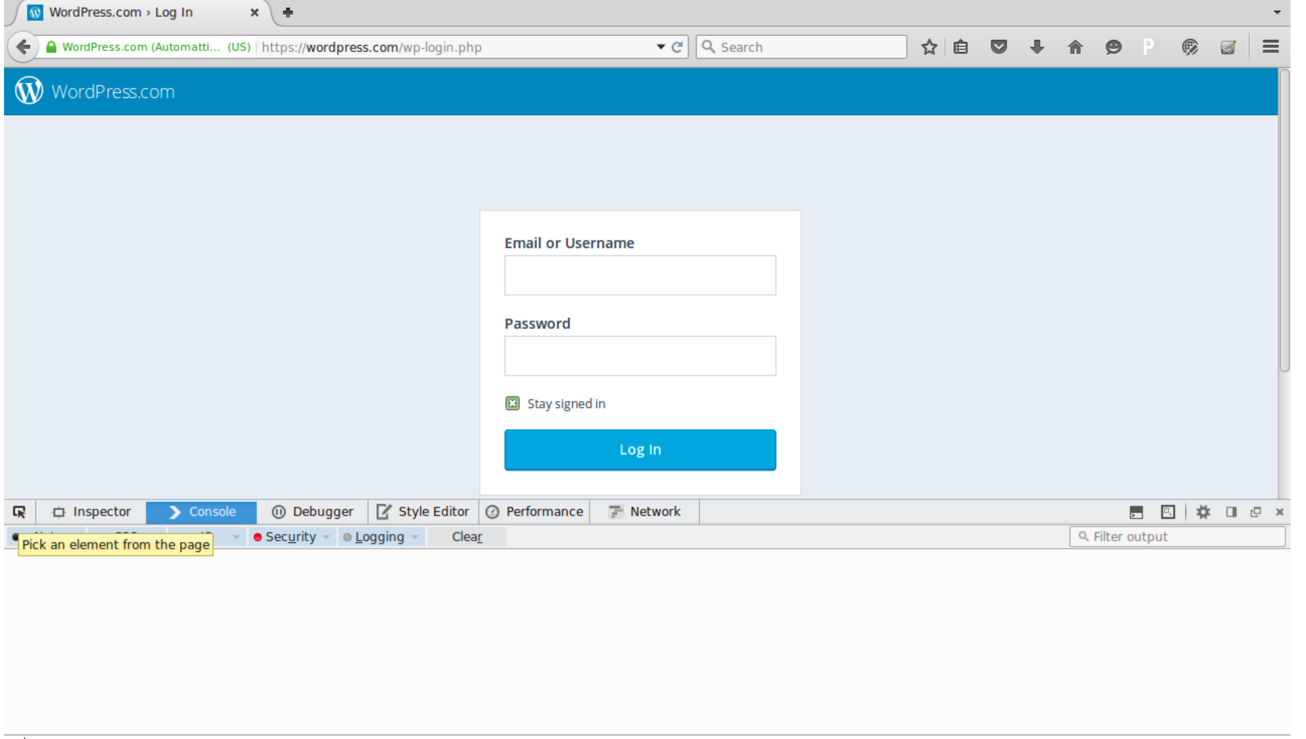
```
File  Edit  View  History  Bookmarks  Tools  Help
WordPress.com › Log In   ×
WordPress.com (Automatti... (US) | ht                    Search
WordPress.com

Tools
  Downloads            Ctrl+Shift+Y
  Add-ons              Ctrl+Shift+A
  Apps
  Set Up Sync...
  Web Developer  ▸
  Page Info            Ctrl+I
  Tab Mix Plus Options...
  Selenium IDE         Ctrl+Alt+S

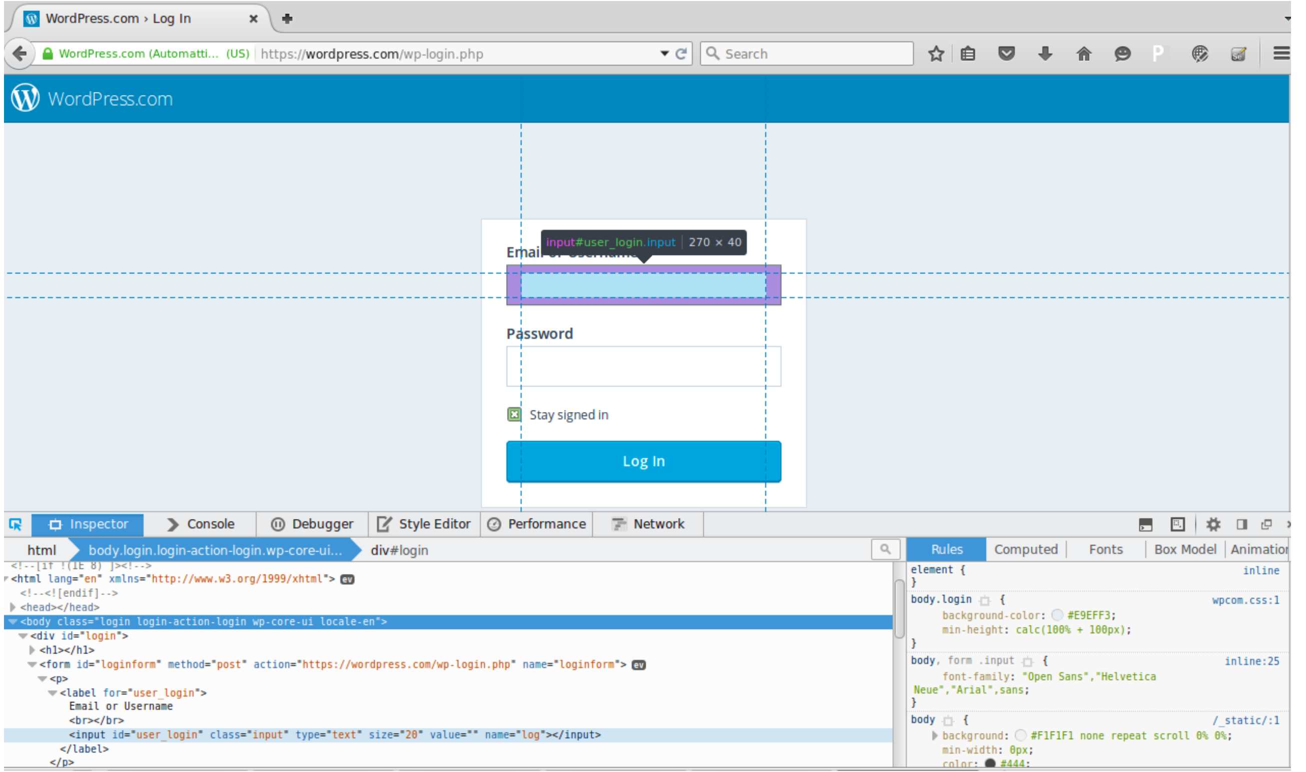
Web Developer
  Toggle Tools            Ctrl+Shift+I
  Inspector               Ctrl+Shift+C
  Web Console             Ctrl+Shift+K
  Debugger                Ctrl+Shift+S
  Style Editor            Shift+F7
  Performance             Shift+F5
  Network                 Ctrl+Shift+Q
  Developer Toolbar       Shift+F2
  WebIDE                  Shift+F8
  Browser Console         Ctrl+Shift+J
  Responsive Design View  Ctrl+Shift+M
  Eyedropper
  Scratchpad              Shift+F4
  Page Source             Ctrl+U
  Get More Tools
  Selenium IDE            Ctrl+Alt+S

Register
```

இது பின்வருமாறு ஒரு விஷயத்தை வலைத்தளப் பக்கத்தின் அடியில் வெளிப்படுத்தும். அங்கு சென்று *'Pick an element from the page'* எனும் *icon*-ன் மீது சொடுக்கவும்.



இனி நாம் வலைத்தளப் பக்கத்தில் சென்று எதன் மீது *cursor*-ஐ வைத்தாலும் அதன் *html code* கீழே தெரியும். எனவே இங்கிருந்து நாம் ஒவ்வொரு *textbox*-ன் *id*-ஐயும் சுலபமாக கண்டுபிடித்துவிடலாம்.



5. அவ்வாறு கண்டுபிடிக்கப்பட்ட username, password textboxes & login buttons-ன் ID களைப் பின்வருமாறு //input[] method-க்குள் கொடுக்க வேண்டும். send_keys எனும் method, username,password-ஐ அந்த textbox-க்குள் செலுத்தவும், click() என்பது buttons-ன் மீது சொடுக்கவும் பயன்பட்டுள்ளது.

```
a.find_element_by_xpath("//input[@id='user_login']").send_keys("valaipathivu")
a.find_element_by_xpath("//input[@id='user_pass']").send_keys("Kadavuchol")
a.find_element_by_xpath("//input[@id='wp-submit']").click()
```

6. இதையடுத்து வரும் print என்பது வெற்றிகரமாக உள்நுழைந்து விட்டோம் என்பதை பயனருக்கு வெளிப்படுத்த உதவுகிறது.

```
print "Login is successful"
```

7. இதைத் தொடர்ந்து கொடுக்கப்பட்டுள்ள பின்வரும் நிரல்கள் அனைத்தும் மேற்கூறிய concept-ஐப் பயன்படுத்தி எழுதப்பட்டவையே ஆகும். எனவே இவற்றை இப்போது நீங்களாகவே புரிந்து கொள்ள முடியும்.

```
a.find_element_by_link_text("Posts").click()
a.find_element_by_link_text("Add New").click()
a.find_element_by_xpath("//input[@id='title']").send_keys("Tamil Kavithaikal")
a.find_element_by_xpath("//input[@id='publish']").click()
a.find_element_by_link_text("Close Sidebar").click()
print "New post is Published"
```

இதன் output பின்வருமாறு

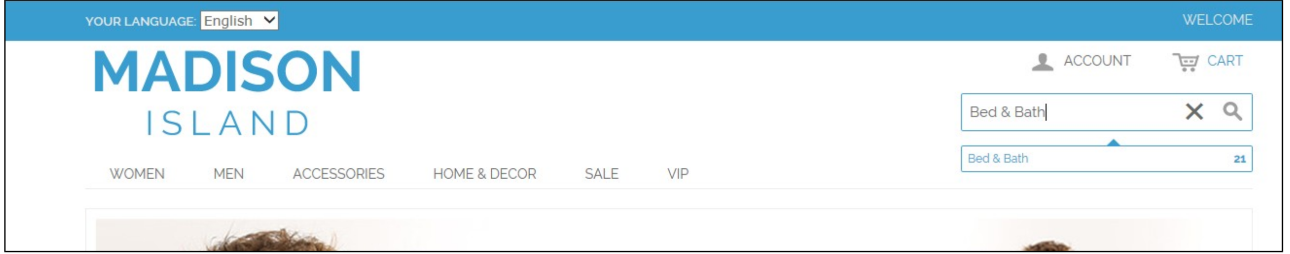
```
nithya@nithya-laptop: ~/Desktop/selenium
File Edit View Search Terminal Help
nithya@nithya-laptop:~/Desktop/selenium$ python wordpress.py
Login is successful
New post is Published
nithya@nithya-laptop:~/Desktop/selenium$
```

இதுபோன்று வலைத்தளப் பக்கங்களின் *html* மொழியுடன் தொடர்பு கொண்டு பல்வேறு வகையான விஷயங்களை நிகழ்த்துவதற்கு *webdriver*-ஆனது பல்வேறு வகையான *methods, attributes, classes* - ஐப் பயன்படுத்துகிறது. அவற்றில் ஒருசில முக்கியமானவைகளைப் பற்றி இந்தப் புத்தகத்தில் காண்போம்.

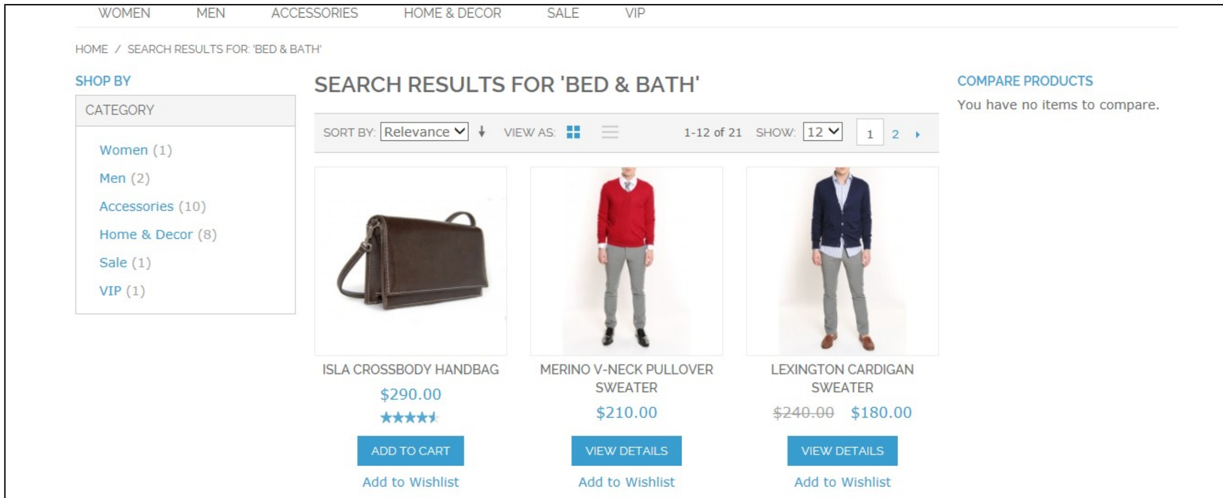
இனிவரும் உதாரணங்கள் அனைத்திற்கும் நான் *http://magento-demo.lexiconn.com/* எனும் வலைத்தளப் பக்கத்தைப் பயன்படுத்தப் போகிறேன்.

3.1 **Search Results**-ஐ வெளிப்படுத்தல்

http://magento-demo.lexiconn.com/



இந்த வலைத்தளத்தின் searchbox-ல் சென்று "Bed & Bath" எனக் கொடுக்கும்போது, அது பின்வருமாறு 12 விடைகளை வெளிப்படுத்துகிறது.



இதை automate செய்வதற்கான code பின்வருமாறு அமையும்.

```
#File : 1-search.py
from selenium import webdriver

a = webdriver.Firefox()
a.get("http://magento-demo.lexiconn.com/")
a.maximize_window()

a.find_element_by_xpath("//input[@id='search']").send_keys("Bed & Bath")
a.find_element_by_xpath("//button[@title='Search']").click()

lis = a.find_elements_by_xpath("//h2[@class='product-name'] / a ")

print str(len(lis)) + " products found"

for i in lis:
      print i.text

a.quit()
```

இந்தக் *code*-ல் உள்ள முதல் *6* வரிகளை இப்போது உங்களால் சுலபமாகப் புரிந்து கொள்ள முடியும்.

அதற்கு அடுத்த வரியான *h2[@class='product-name'] / a* என்பது *result*-ல் *(class = 'product-name')* உள்ள அனைத்து இணைப்புகளையும் *(<a>)* தேர்ந்தெடுத்து *lis* எனும் *variable*-ல் பட்டியலாக சேமிக்கிறது. பின்னர் அவை *for loop* வழியாக ஒவ்வொன்றாக வெளிப்படுத்தப்படுகின்றன.

print str(len(lis)) என்பது *result*-ல் மொத்தம் எத்தனை இணைப்புகள் உள்ளது என்பதைக் கணக்கிட்டு வெளிப்படுத்துகிறது. அதாவது *len()* மூலம் பட்டியலில் உள்ள இணைப்புகளின் எண்ணிக்கை கணக்கிடப்படுகிறது. பின்னர் ஒரு எண்ணை *print* செய்ய முடியாது என்பதால் அந்த மதிப்பு *str()* மூலம் மாற்றப்பட்டு *print* செய்யப்படுகிறது. இதன் *output* பின்வருமாறு இருக்கும்.

```
nithya@nithya-laptop: ~/Desktop/selenium/Programs
File  Edit  View  Search  Terminal  Help
nithya@nithya-laptop:~/Desktop/selenium/Programs$ python 1-search.py
12 products found
ISLA CROSSBODY HANDBAG
MERINO V-NECK PULLOVER SWEATER
LEXINGTON CARDIGAN SWEATER
DELANCY CARDIGAN SWEATER
DORIAN PERFORATED OXFORD
CLASSIC HARDSHELL SUITCASE
LUGGAGE SET
PILLOW AND THROW SET
PEARL STUD EARRINGS
GRAMERCY THROW
PARK ROW THROW
CARNEGIE ALPACA THROW
nithya@nithya-laptop:~/Desktop/selenium/Programs$
```

3.2 **assertEqual**() மூலம் ஒப்பீடுகள் செய்தல்

மேற்கண்ட program-ல் "Bed & Bath" எனக் கொடுத்து தேடும்போது, அது வெளிப்படுத்தும் இணைப்புகளின் எண்ணிக்கையை எவ்வாறு கணக்கிடுவது மற்றும் வெளிப்படுத்துவது எனப் பார்த்தோம்.

இந்தப் பகுதியில் "Bed & Bath" எனக் கொடுத்து தேடும்போது, வலைத்தளமானது 12 இணைப்புகளை மட்டுமே வெளிப்படுத்த வேண்டும் என்று ஒரு condition இருந்தால் அதை எவ்வாறு program-ல் சோதிப்பது என்று பார்க்கப் போகிறோம். இதற்கு assertEqual() function பயன்படும்.

```
#File : 2-assersions.py

import unittest
from selenium import webdriver

class SearchTests (unittest.TestCase):
    def setUp(self):
        self.a = webdriver.Firefox()
        self.a.get("http://magento-demo.lexiconn.com/")
        self.a.maximize_window()

    def test_search(self):
      self.a.find_element_by_xpath("//input[@id='search']").send_keys("Bed & Bath")
      self.a.find_element_by_xpath("//input[@id='search']").submit()
      lis = self.a.find_elements_by_xpath("//h2[@class='product-name']/a")
      self.assertEqual(12, len(lis))

    def tearDown(self):
      self.a.close()

if __name__ == '__main__':
      unittest.main(verbosity=2)
```

இந்த assertEqual() - ஐ மட்டும் நம்மால் தனியாகப் பயன்படுத்த முடியாது. ஏனெனில் இது unittest library-க்குள் TestCase எனும் class-க்குள் வரையறுக்கப்பட்டுள்ளது. எனவே நமது program-ன் முதல் வரியாக unittest-ஆனது import செய்யப்பட்டுள்ளது.

பின்னர் நாம் வரையறுத்துள்ள SearchTests க்குள் நம்மால் assertEqual-ஐப் பயன்படுத்த முடியும். ஏனெனில் இதற்குத் தேவையான TestCase class-ஆனது இங்கு argument-ஆக செலுத்தப்பட்டுள்ளது.

setUp() மற்றும் tearDown() ஆகியவை ஒரு புதிய browser-ஐ திறந்து மூடப்

பயன்படுகின்றன. இந்த இரண்டு *methods*-ம் *unittest* இருக்கின்ற அனைத்து நிரல்களிலும் பயன்படுத்தப்பட வேண்டும்.

இந்த இரண்டிற்கும் நடுவில் நாம் உருவாக்கப்போகும் அனைத்து *functions*-ம் *test_*என்றே தொடங்க வேண்டும். அவ்வாறு உருவாக்கப்பட்ட *test_search()*-ல் தான் வலைத்தளமானது *12* இணைப்புகளை மட்டுமே வெளிப்படுத்துகிறதா எனும் சோதனை செய்யப்பட்டுள்ளது. இதற்கான நிரல்களை படித்தாலே உங்களுக்குப் புரிந்துவிடும்.

ஒவ்வொரு *method*-க்குள்ளும் உள்ள அனைத்து வரிகளும் *self.* என தொடங்கப்பட்டுள்ளத்தை கவனிக்கவும் (*variable* தவிர). இது இப்படித்தான். எனவே கேள்வி கேட்காமல் நீங்களும் அப்படியே பயன்படுத்தவும். அவ்வாறே கடைசியாக உள்ள *if __name* எனத் தொடங்கும் வரி நாம் உருவாக்கிய *program*-ஐ அழைத்து *run* செய்யப் பயன்படும். எனவே கேள்வி கேட்காமல் அதையும் கடைசியில் இணைத்துவிடவும். இதற்கான *output* பின்வருமாறு.

```
nithya@nithya-laptop: ~/Desktop/selenium/Programs
File  Edit  View  Search  Terminal  Help
nithya@nithya-laptop:~/Desktop/selenium/Programs$ python 2-assertions.py
test_search (__main__.SearchTests) ... ok

----------------------------------------------------------------------
Ran 1 test in 17.587s
```

ஒருவேளை நாம் நமது *program*-ன் *assertEqual*-க்குள் *20* எனக் கொடுத்துள்ளோம். ஆனால் உண்மையில் வலைத்தளமானது *12* இணைப்புகளை மட்டுமே வெளிப்படுகிறதெனில், *program*-ஆனது பயனருக்கு *AssertionError* என்பதை வெளிப்படுத்தும். அதாவது நாம் கொடுத்துள்ளபடி இணைப்புகள் வெளிப்படவில்லை என்று இதற்கு அர்த்தம். இது பின்வருமாறு.

```
nithya@nithya-laptop: ~/Desktop/selenium/Programs
File  Edit  View  Search  Terminal  Help
nithya@nithya-laptop:~/Desktop/selenium/Programs$ python 2-assertions.py
test_search (__main__.SearchTests) ... FAIL

======================================================================
FAIL: test_search (__main__.SearchTests)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "2-assertions.py", line 14, in test_search
    self.assertEqual(20, len(lis))
AssertionError: 20 != 12

----------------------------------------------------------------------
Ran 1 test in 17.207s

FAILED (failures=1)
nithya@nithya-laptop:~/Desktop/selenium/Programs$
```

3.3 @classmethod பயன்படுத்தாவிடில் நிகழுவது

மேற்கண்ட program-ல் "Bed & Bath" எனக் கொடுத்து தேடும்போது அது வெளிப்படுத்தும் இணைப்புகளின் மதிப்பு 12-ஆக உள்ளதா என ஒரு method-க்குள் சோதித்தோம்.

இப்போது இந்த program-ல் இன்னொரு method- ஐ உருவாக்கி அதற்குள் "Bags & Luggage" எனக் கொடுத்து தேடும்போது அது 11 இணைப்புகளை வெளிப்படுத்துகிறதா என்பதை சோதிக்கப் போகிறோம்.

```
File : 3-multi-assersion.py

import unittest
from selenium import webdriver

class SearchTests (unittest.TestCase):
    def setUp(self):
        self.a = webdriver.Firefox()
        self.a.get("http://magento-demo.lexiconn.com/")
        self.a.maximize_window()

    def test_search_product1(self):
      self.a.find_element_by_xpath("//input[@id='search']").send_keys("Bed &
Bath")
      self.a.find_element_by_xpath("//input[@id='search']").submit()
      lis = self.a.find_elements_by_xpath("//h2[@class='product-name']/a")
      self.assertEqual(12, len(lis))

    def test_search_product2(self):
      self.a.find_element_by_xpath("//input[@id='search']").send_keys("Bags &
Luggage")
      self.a.find_element_by_xpath("//input[@id='search']").submit()
      lis = self.a.find_elements_by_xpath("//h2[@class='product-name']/a")
      self.assertEqual(12, len(lis))

    def tearDown(self):
      self.a.close()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

இந்த program-ஐ run செய்யும்போது test_search_product1 எனும் முதல் சோதனைக்கு ஒருமுறை browser-ஐ திறந்து சோதனைகளை நிகழ்த்தி மூடிவிடும். அவ்வாறே test_search_product2 எனும் இரண்டாவது சோதனைக்கும் ஒருமுறை browser-ஐ திறந்து சோதனைகளை நிகழ்த்தி மூடிவிடும்.

அதாவது இந்த *program*-க்குள் வரையறுக்கப்பட்டுள்ள ஒவ்வொரு தனித்தனி சோதனைகளுக்கும் *browser*-ஆனது ஒவ்வொரு முறை தனித்தனியாக திறந்து திறந்து மூடப்படும். எனவே சோதனைகளை நிகழ்த்தி முடிப்பதற்கான நேரம் அதிகமாகிறது. இதற்கான பின்வரும் *output*- ல் சோதனைகள் அனைத்தும் *33.360s* கால அளவில் நிகழ்த்து முடிந்திருப்பதைக் காணலாம்.

```
nithya@nithya-laptop: ~/Desktop/selenium/Programs
File  Edit  View  Search  Terminal  Help
nithya@nithya-laptop:~/Desktop/selenium/Programs$ python 3-multi_assertion.py
test_search_product1 (__main__.SearchTests) ... ok
test_search_product2 (__main__.SearchTests) ... ok

----------------------------------------------------------------------
Ran 2 tests in 33.360s

OK
nithya@nithya-laptop:~/Desktop/selenium/Programs$
```

இவ்வாறு தனித்தனியாக *browsers* திறந்து மூடப்படுவதையும், அதனால் ஏற்படும் கால விரயத்தையும் தவிர்ப்பதற்காக வந்ததே *@classmethod* ஆகும்.

3.4 @**classmethod**-ன் பயன்பாடு

நமது program-ல் browser-ஐ திறப்பதற்கு முன்னர் @classmethod எனக் கொடுத்து நாம் என்னென்ன செய்ய விரும்புகிறோமோ அதற்கான methods-ஐ எல்லாம் வரையறுக்க வேண்டும். பின்னர் கடைசியாக மீண்டும் @classmethod எனக் கொடுத்து browser-ஐ மூட வேண்டும். இதுவே அனைத்து methods-ஐயும் நாம் விரும்பும் வகையில் தொடர்ச்சியாக நிகழ்த்த உதவும்.

```
#File : search_tests.py

import unittest
from selenium import webdriver

class SearchTests (unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        cls.a = webdriver.Firefox()
        cls.a.get("http://magento-demo.lexiconn.com/")
        cls.a.maximize_window()

    def test_search_product1(self):
        self.a.find_element_by_xpath("//input[@id='search']").send_keys("Bed &
Bath")
        self.a.find_element_by_xpath("//input[@id='search']").submit()
        lis = self.a.find_elements_by_xpath("//h2[@class='product-name']/a")
        self.assertEqual(12, len(lis))

    def test_search_product2(self):
        self.a.find_element_by_xpath("//input[@id='search']").clear()
        self.a.find_element_by_xpath("//input[@id='search']").send_keys("Bags &
Luggage")
        self.a.find_element_by_xpath("//input[@id='search']").submit()
        lis = self.a.find_elements_by_xpath("//h2[@class='product-name']/a")
        self.assertEqual(12, len(lis))
    @classmethod

    def tearDownClass(cls):
      cls.a.quit()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

இதற்கான பின்வரும் output- ல் சோதனைகள் அனைத்தும் 18.461s கால அளவில் நிகழ்த்து முடிந்திருப்பதைக் காணலாம். அதாவது @classmethod பயன்படுத்தாதபோது, இது எடுத்துக்கொண்ட கால அளவைவிட இப்போது குறைவாகவே எடுத்துக்கொண்டுள்ளது.

```
nithya@nithya-laptop: ~/Desktop/selenium/Programs
File  Edit  View  Search  Terminal  Help
nithya@nithya-laptop:~/Desktop/selenium/Programs$ python search_tests.py
test_search_product1 (__main__.SearchTests) ... ok
test_search_product2 (__main__.SearchTests) ... ok

----------------------------------------------------------------------
Ran 2 tests in 18.461s

OK
nithya@nithya-laptop:~/Desktop/selenium/Programs$
```

3.5 ஒரு பக்கத்தில் இருக்க வேண்டியவற்றை சோதித்தல்

http://magento-demo.lexiconn.com/ எனும் வலைத்தளப்பக்கத்தில் இருக்க வேண்டிய பின்வரும் முக்கிய விஷயங்களை சோதிப்பதற்கான நிரல் அதற்குக் கீழே கொடுக்கப்பட்டுள்ளது.

1. தேடும் பெட்டி
2. மொழிகளின் தேர்வுப் பட்டியல்
3. கைக்கூடை
4. எனது கணக்கு
5. விளம்பரப் படங்கள்

```
#File : homepage-tests.py


import unittest
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.by import By
from __builtin__ import classmethod

class HomePageTest (unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        cls.a = webdriver.Firefox()
        cls.a.get("http://magento-demo.lexiconn.com/")
        cls.a.maximize_window()

    def test_searchbox(self):
        self.assertTrue (self.is_element_present (By.ID,"search"))
        self.assertTrue (self.a.find_element_by_id("search").is_enabled())
      x = self.a.find_element_by_id("search").get_attribute("maxlength")
      self.assertEqual("128",x)

    def test_languagebox(self):
        self.assertTrue (self.is_element_present (By.ID,"select-language"))

    def test_addcart(self):
      self.a.find_element_by_css_selector("div.header-minicart 
span.icon").click()
      x = "You have no items in your shopping cart."
      y = self.a.find_element_by_css_selector("p.empty").text
      self.assertEqual(x,y)
      self.a.find_element_by_css_selector("div.minicart-wrapper 
a.close").click()

    def test_account(self):
      x = self.a.find_element_by_link_text("ACCOUNT")
      y = self.a.find_elements_by_partial_link_text("ACCOUNT")
```

```
      self.assertTrue(x.is_displayed())
      self.assertTrue(2,len(y))

    def test_imgs(self):
      x = self.a.find_element_by_class_name("promos")
      y = x.find_elements_by_tag_name("img")
      self.assertEqual(3,len(y))

      z = self.a.find_element_by_xpath("//img[@alt='Shop Private Sales - Members
Only']")
      self.assertTrue(z.is_displayed())
      z.click()
      self.assertEqual("VIP",self.a.title)

    @classmethod
    def tearDownClass(cls):
      cls.a.quit()

    def is_element_present(self, how, what):
        try: self.a.find_element(by=how, value=what)
      except NoSuchElementException, e: return False
      return True

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

இதன் *output* மற்றும் நிரல்களுக்கான விளக்கத்தைப் பின்வருமாறு காணலாம்.

```
nithya@nithya-laptop: ~/Desktop/selenium/Programs
File  Edit  View  Search  Terminal  Help
nithya@nithya-laptop:~/Desktop/selenium/Programs$ python homepage_tests.py
test_account (__main__.HomePageTest) ... ok
test_addcart (__main__.HomePageTest) ... ok
test_imgs (__main__.HomePageTest) ... ok
test_languagebox (__main__.HomePageTest) ... ok
test_searchbox (__main__.HomePageTest) ... ok

----------------------------------------------------------------------
Ran 5 tests in 19.430s

OK
nithya@nithya-laptop:~/Desktop/selenium/Programs$
```

3.5.1. தேடும் பெட்டி



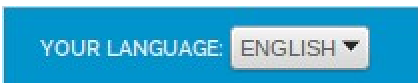
search எனும் பெட்டி உள்ளதா? அது பயனர் பயன்படுத்தும் வகையில் *enable*-ஆக உள்ளதா? அந்த பெட்டிக்குள் *128* எழுத்துக்களை செலுத்த முடியுமா? என்பது போன்ற விஷயங்களெல்லாம் *test_searchbox()*-க்குள் சோதிக்கப்பட்டுள்ளது.

assertTrue() என்பது

- ஒரு *element* உள்ளதா என்பதை சோதிப்பதற்கு *.is_element_present()* ஐயும்
- அது *enable* ஆக உள்ளதா என்பதை சோதிப்பதற்கு *.is_enabled()* ஐயும் பயன்படுத்தியுள்ளது.

மேலும் *.get_attribute()* ஐப் பயன்படுத்தி அந்த பெட்டியின் நீளம் கண்டுபிடிக்கப்பட்டு *x* என வைக்கப்பட்டுள்ளது. பின்னர் *assertEqual()* மூலம் *x*-ஆனது *128*-உடன் ஒப்பிடப்பட்டு, அந்தப் பெட்டி *128* எழுத்துக்களைத் தாங்கும் வகையில் உருவாக்கப்பட்டுள்ளதா என்பது சோதிக்கப்பட்டுள்ளது.

3.5.2. மொழிகளின் தேர்வுப் பட்டியல்

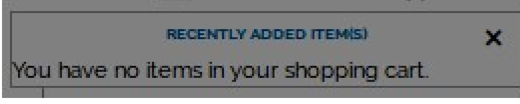


நாம் விரும்பும் மொழியினைத் தேர்ந்தெடுக்கும் பட்டியலைக் கொண்ட பெட்டி உள்ளதா என்பது *test_languagebox()*-க்குள் சோதிக்கப்பட்டுள்ளது.

assertTrue() என்பது இந்த *element* உள்ளதா என்பதை சோதிப்பதற்கு *.is_element_present()*-ஐப் பயன்படுத்தியுள்ளது.

3.5.3. கைக்கூடை





CART icon க்கு சென்று அதன்மீது சொடுக்கும்போது, *"You have no items in your shopping cart"* என்று வருகிறதா என்பது *test_addcart()*-க்குள் சோதிக்கப்பட்டுள்ளது.

CART icon க்கு சென்று அதன்மீது சொடுக்கவும் என்பதற்கான முதல் வரி *find_element_by_css_selector("div.header-minicart span.icon")* என்று உள்ளது. இந்த *icon*- க்கான நிரல் *<div>* -ஐக் கொண்டிருப்பதால் இதனை நாம் *css* கொண்டு அணுக வேண்டும்.

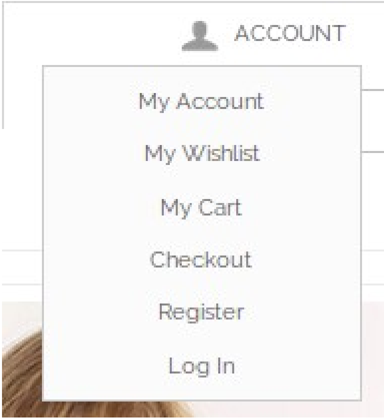
```
<div class="header-minicart">
  <a class="skip-link skip-cart no-count" href="#header-cart"> ev
    <span class="icon"></span>
    <span class="label">Cart</span>
    <span class="count">0</span>
  </a>
  <div id="header-cart" class="block block-cart skip-content"></div> ev
```

எனவே *div*.அதன் *class* ஓர் இடைவெளி விட்டு அதற்குள் உள்ள *span*. அதன் *class* எனக் கொடுக்கப்பட்டுள்ளது. இதுவே ஒரு *css element*-ஐக் கண்டுபிடிப்பதற்கான முறை ஆகும்.

பின்னர் *"You have no items in your shopping cart"* என்பது *x* – லும், *icon*-ஐ சொடுக்கும்போது வரும் வாக்கியம், *.text* மூலம் எடுக்கப்பட்டு *y* - லும் சேமிக்கப்பட்டுள்ளது. பின்னர் *assertEqual()* மூலம் *x,y* சமமாக உள்ளதா என்பது சோதிக்கப்படுகிறது.

கடைசியாக நமக்கு தகவல் காட்டுகின்ற பெட்டியை மூடுவதற்கான நிரல் கொடுக்கப்பட்டுள்ளது.

3.5.4. எனது கணக்கு



பின்வரும் விஷயங்கள் *test_account()*-க்குள் சோதிக்கப்பட்டுள்ளது.

- *ACCOUNT* எனும் இணைப்பு உள்ளதா?
- *ACCOUNT, My Account* எனும் 2 இணைப்புகள் உள்ளனவா?

இதற்கான நிரலில் *ACCOUNT* எனும் வார்த்தையை மட்டும் கொண்ட இணைப்பு *x*-லும், அந்த வார்த்தையில் தொடங்கும் அல்லது முடியும் எல்லா இணைப்புகளும் *y*-லும் சேமிக்கப்பட்டுள்ளது. பின்னர் *.is_displayed()* மூலம் அந்த இணைப்பு தெரிகிறதா என்பதும், *assertTrue()* மூலம் அந்த வார்த்தையைக் கொண்ட 2 இணைப்புகள் *(Account, My Account)* உள்ளதா என்பதும் பரிசோதிக்கப்பட்டுள்ளன.

3.5.5. விளம்பரப் படங்கள்

பின்வரும் 3 விளம்பரப் படங்கள் தென்படுகின்றனவா மேலும் நடுவில் உள்ள *Shop Private Sales - Members Only* எனும் படத்தின்மீது சொடுக்கும்போது *VIP* எனும் பெரிய படம் வருகிறதா என்பது போன்ற விஷயங்களெல்லாம் *test_imgs()*-க்குள் சோதிக்கப்பட்டுள்ளது.

HOME & DECOR
FOR ALL YOUR SPACES

SHOP PRIVATE SALES
MEMBERS ONLY

TRAVEL GEAR
FOR EVERY OCCASION

VIP



SORT BY: Position  VIEW AS:  3 Item(s) SHOW: 12

பின்வரும் நிரலில் பார்த்தால் அந்த 3 படங்களும் *promos* எனும் *unordered list*-க்குள் உள்ளன. எனவேதான் *find_element_by_class_name* மூலம் *promos*-ல் உள்ள 3 படங்களும் *x* என சேமிக்கப்பட்டுள்ளன. பின்னர் *len()* மூலம் *x*-ன் நீளம் கண்டுபிடிக்கப்பட்டு அது 3 ஆக உள்ளதா என்பது சோதிக்கப்படுகிறது.

```
<ul class="promos">
  <li>
    <a href="http://demo.magentocommerce.com/home-decor.html">
      <img alt="Physical & Virtual Gift Cards" src="http://demo.magentocommerce.com/media/wysiwyg/homepage-three-column-promo-01B.png"></img>
    </a>
  </li>
  <li>
    <a href="http://demo.magentocommerce.com/vip.html">
      <img alt="Shop Private Sales - Members Only" src="http://demo.magentocommerce.com/media/wysiwyg/homepage-three-column-promo-02.png"></img>
    </a>
  </li>
```

3.6 தனித்தனி சோதனைகளை ஒன்றாக இணைத்து இயக்குதல்

search_tests (3.4) மற்றும் *homepage_tests (3.5)* எனும் இரண்டு வெவ்வேறு *program*-களில் கொடுக்கப்பட்டுள்ள சோதனைகளை ஒருசேர நிகழ்த்த விரும்பினால் அதற்கான *code* பின்வருமாறு அமையும்.

```
#File : 6-combining-tests.py

import unittest
from search_tests import SearchTests
from homepage_tests import HomePageTest

x = unittest.TestLoader().loadTestsFromTestCase(SearchTests)
y = unittest.TestLoader().loadTestsFromTestCase(HomePageTest)
z = unittest.TestSuite([x, y])
unittest.TextTestRunner(verbosity=2).run(z)
```

இதில் தேவையான கோப்புகளிலிருந்து அதிலுள்ள *classes*-ன் பெயர்கள் ஏற்றுமதி செய்யப்பட்டிருப்பதைக் காணலாம். பின்னர் அந்த *classes*-ல் உள்ளவை அனைத்தும் *x , y* எனும் *variables*-க்கு செலுத்தப்பட்டுள்ளது. *Z* எனும் *variable*-ஆனது *x* மற்றும் *y*-ஐ இணைத்த ஒரு *TestSuit*-ஐத் தாங்கியுள்ளது. கடைசி வரியானது அனைத்தையும் உள்ளடக்கிய *z*-ஐ இயக்கப் பயன்படுகிறது.

இதற்கான *output* பின்வருமாறு இரண்டு தனித்தனி கோப்புகளில் உள்ள அனைத்து *methods*-ஐயும் ஒருசேர இயக்கி வெளிப்படுத்தும்.

```
nithya@nithya-laptop: ~/Desktop/selenium/Programs
File  Edit  View  Search  Terminal  Help
nithya@nithya-laptop:~/Desktop/selenium/Programs$ python 6-combining_tests.py
test_search_product1 (search_tests.SearchTests) ... ok
test_search_product2 (search_tests.SearchTests) ... ok
test_account (homepage_tests.HomePageTest) ... ok
test_addcart (homepage_tests.HomePageTest) ... ok
test_imgs (homepage_tests.HomePageTest) ... ok
test_languagebox (homepage_tests.HomePageTest) ... ok
test_searchbox (homepage_tests.HomePageTest) ... ok

----------------------------------------------------------------------
Ran 7 tests in 42.655s

OK
nithya@nithya-laptop:~/Desktop/selenium/Programs$
```

3.7 **Report** உருவாக்குதல்

மேற்கண்ட (3.6) அதே சோதனை முடிவுகளை ஒரு HTML Report-ஆக மாற்ற விரும்பினால் அதற்கான நிரல் பின்வருமாறு அமையும்.

```
#File : 7-test-report.py

import unittest
import HTMLTestRunner
import os
from search_tests import SearchTests
from homepage_tests import HomePageTest

dir = os.getcwd()

x = unittest.TestLoader().loadTestsFromTestCase(SearchTests)
y = unittest.TestLoader().loadTestsFromTestCase(HomePageTest)
z = unittest.TestSuite([x, y])
outfile = open("SmokeTestReport.html", "w")
r = HTMLTestRunner.HTMLTestRunner(stream=outfile,title='Test
Report',description='Smoke Tests')
r.run(z)
```

இதில் உள்ள x,y,z,r ஆகியவை அனைத்தும் variables ஆகும். மீதி அனைத்தையும் இந்நிலையில் உங்களாலேயே புரிந்து கொள்ள முடியும் என்று நினைக்கிறேன்.

இதை run செய்வதற்கு முன்னர் http://tungwaiyip.info/software/HTMLTestRunner.html எனும் முகவரிக்கு சென்று HTMLTestRunner.pyc எனும் கோப்பினை இறக்குமதி செய்து நமது கோப்புகள் சேமிக்கப்பட்டுள்ள அதே Folder-க்குள் இதையும் சேமித்து விடவும். அப்போதுதான் நமது program எந்த தவறும் இல்லாமல் இயங்கும்.



இதன் output பின்வருமாறு வெளிப்பட்டாலும், நமது கோப்புகள் அனைத்தும் சேமிக்கப்பட்டுள்ள folder-க்குள் நாம் குறிப்பிட்டிருக்கும் பெயரில் ஒரு html report-ஐ உருவாக்கும்.

```
nithya@nithya-laptop: ~/Desktop/selenium/Programs
File Edit View Search Terminal Help
nithya@nithya-laptop:~/Desktop/selenium/Programs$ python 7-testreport.py
.......
Time Elapsed: 0:00:36.384509
nithya@nithya-laptop:~/Desktop/selenium/Programs$
```



SmokeTestReport.html

இதனை திறந்து பார்த்தால் பின்வருமாறு ஒரு *report* வெளிப்படும்.

file:///home/nithya/Desktop/selenium/Programs/SmokeTestReport.html

Test Report

Start Time: 2015-09-11 13:34:59
Duration: 0:00:36.384509
Status: Pass 7

Smoke Tests

Show Summary Failed All

Test Group/Test case	Count	Pass	Fail	Error	View
search_tests.SearchTests	2	2	0	0	Detail
test_search_product1			pass		
test_search_product2			pass		
homepage_tests.HomePageTest	5	5	0	0	Detail
test_account			pass		
test_addcart			pass		
test_imgs			pass		
test_languagebox			pass		
test_searchbox			pass		
Total	**7**	**7**	**0**	**0**	

3.8 படிவம் பூர்த்தி செய்தல்

Magento தளத்தில் சென்று *Account -> Login -> Create an Account* என்பதன் மீது சொடுக்கினால் வரும் படிவத்தை *automate* செய்வதற்கான நிரல் பின்வருமாறு அமையும்.

```
File : 8-registration-form.py

import unittest
from selenium import webdriver
from time import gmtime,strftime

class Login (unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        cls.a = webdriver.Firefox()
        cls.a.implicitly_wait(30)
        cls.a.get("http://magento-demo.lexiconn.com/")
        cls.a.maximize_window()

    def test_register(self):
        self.a.find_element_by_link_text("ACCOUNT").click()
        self.a.find_element_by_link_text("Log In").click()

      x = self.a.find_element_by_link_text("CREATE AN ACCOUNT")
      self.assertTrue(x.is_displayed() and x.is_enabled())
      x.click()
      self.assertEquals("Create New Customer Account",self.a.title)
```

```
      fn = self.a.find_element_by_id("firstname")
      ln = self.a.find_element_by_id("lastname")
      ea = self.a.find_element_by_id("email_address")
      pa = self.a.find_element_by_id("password")
      co = self.a.find_element_by_id("confirmation")
      su = self.a.find_element_by_id("is_subscribed")
      re = self.a.find_element_by_xpath("//button[@title='Register']")

      self.assertEqual("255", fn.get_attribute("maxlength"))
      self.assertEqual("255", ln.get_attribute("maxlength"))
      self.assertTrue(fn.is_enabled() and ln.is_enabled() and ea.is_enabled()
and pa.is_enabled() and
      co.is_enabled() and su.is_enabled() and re.is_enabled())
      self.assertFalse(su.is_selected())

      laname = "user_" + strftime("%Y%m%d%H%M%S", gmtime())
      fn.send_keys("Test")
      ln.send_keys(laname)
      su.click()
      ea.send_keys(laname + "@example.com")
      pa.send_keys("tester")
      co.send_keys("tester")
      re.click()

      self.assertEqual("Hello, Test " + laname +
"!",self.a.find_element_by_css_selector("p.hello >strong").text)
      self.a.find_element_by_link_text("ACCOUNT").click()
      self.assertTrue(self.a.find_element_by_link_text("Log
Out").is_displayed())

    @classmethod
    def tearDownClass(cls):
      cls.a.quit()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

test_register எனும் method-க்குள் என்னென்ன எழுதியுள்ளேன் என்பதைப் பின்வருமாறு காணலாம்.

1. *முதலில் Account, Log In எனும் இணைப்பின் மீது சென்று சொடுக்குவதற்கான நிரல் கொடுக்கப்பட்டுள்ளது.*
2. *"Create an Account" எனும் இணைப்பினை x என வைத்து, அது உள்ளதா மற்றும் enable-ஆக உள்ளதா என்பது assertTrue-க்குள் சோதிக்கப்பட்டுள்ளது. பின்னர் இதன் மீது சொடுக்கும்போது, "Create New Customer Account" எனும் தலைப்பினை வெளிப்படுத்துகிறதா என்பது assertEquals-க்குள் சோதிக்கப்பட்டுள்ளது.*
3. *இந்தப் படிவத்தில் பெறப்படுகின்ற ஒவ்வொரு விவரமும் fn, ln, ea .... என்பது போன்ற variables-ல் சேமிக்கப்படுகின்றன. இத்தகைய*

variables அனைத்தும் படிவத்தில் உள்ளதா மற்றும் அவை enable-ஆக உள்ளதா என்பது assertTrue-க்குள் சோதிக்கப்படுகிறது.

4. மேலும் firstname, lastname பெறுவதற்கான பெட்டியானது 255 எழுத்துக்கள் வரை தாங்கும் வகையில் உள்ளதா என்பது assertEqual மூலம் சோதிக்கப்பட்டுள்ளது.
5. கடைசியாக உள்ள checkbox-ஆனது tick செய்யப்படாமல் இருக்கிறதா என்பது assertFalse(su.is_selected()) எனக் கொடுத்து சோதிக்கப்பட்டுள்ளது.
6. பின்னர் படிவத்தில் உள்ள ஒவ்வொரு பெட்டியினுள்ளும் என்னென்ன விவரங்களை செலுத்த வேண்டுமோ அவை send_keys மூலம் செலுத்தப்பட்டுள்ளன. Checkbox மற்றும் button-ன் மீது சொடுக்க வேண்டும் என்பதற்கு அவற்றைத் தாங்கியிருக்கும் variable-ஐத் தொடர்ந்து click () எனக் கொடுக்கப்பட்டுள்ளது.
7. ஒவ்வொரு முறையும் ஒரே மாதிரியான தகவல்களை கொடுத்து விண்ணப்பிக்க முடியாது என்பதற்காக laname variable-ல் User_தற்போதையநேரம் சேமிக்கப்பட்டு, அது lastname-ஆகவும் மின்னஞ்சல் முகவரியாகவும் செலுத்தப்பட்டுள்ளது. strftime மூலம் தற்போதைய நேரம் எடுக்கப்பட்டுள்ளது. இதற்குத் தேவையான modules தொடக்கத்தில் import செய்யப்பட்டிருப்பதைக் காணலாம்.
8. ஒரு பயனர் வெற்றிகரமாக விண்ணப்பிக்கப்பட்ட பின்னர் வெளிப்படும் செய்தியானது, css selector மூலம் எடுக்கப்பட்டு, அது நாம் எதிர்பார்க்கும் விதத்தில் உள்ளதா என்பது assertEquals- மூலம் சோதிக்கப்பட்டுள்ளது.
9. கடைசியாக Account எனும் இணைப்பின் மீது சொடுக்கி அதற்குள் Log out உள்ளதா என்பது சோதிக்கப்பட்டுள்ளது.

என்னென்ன வகையான assertions உள்ளது என்பதைப் பின்வருமாறு காணலாம். மேற்கண்ட அதே program-ல் assert-க்கான வரியை மட்டும் மாற்றி மாற்றி பின்வருமாறு கொடுக்கவும்.

1. assertEquals (a,b) - இது கொடுக்கப்பட்டுள்ள இரண்டு வார்த்தைகளும் சமமாக உள்ளதா என்பதை ஒப்பிடும்.

```
self.assertEquals("Create New Customer Account",self.a.title)
```

சமமாக இருப்பின் பின்வருமாறு output - ஐ வெளிப்படுத்தும்.

```
nithya@nithya-laptop:~/Desktop/selenium/Programs$ python Assertions.py
test_register (__main__.Login) ... ok

----------------------------------------------------------------------
Ran 1 test in 23.341s

OK
```

2. assertNotEqual (a,b) - இது கொடுக்கப்பட்டுள்ள இரண்டு வார்த்தைகளும் சமமாக இல்லையா என்பதை சோதிக்கும்.

```
self.assertNotEqual("Create New Customer Account",self.a.title)
```

அவை சமமாக இருக்கும் பட்சத்தில் பின்வருமாறு error- ஐ வெளிப்படுத்தும்.

```
nithya@nithya-laptop:~/Desktop/selenium/Programs$ python Assertions.py
test_register (__main__.Login) ... FAIL

======================================================================
FAIL: test_register (__main__.Login)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "Assertions.py", line 21, in test_register
    self.assertNotEqual("Create New Customer Account",self.a.title)
AssertionError: 'Create New Customer Account' == u'Create New Customer Account'

----------------------------------------------------------------------
Ran 1 test in 15.286s
```

3. assertEqual (a,b,[msg]) - இது இரண்டு வார்த்தைகளும் சமமாக உள்ளதா என சோதிக்கும். அவ்வாறு சமமாக இல்லையெனில் Hai-ஐ வெளிப்படுத்தும்.

```
self.assertEqual("Create New Customer Account",self.a.title,["Hai"])
```

இங்கு இரண்டும் சமமாக உள்ளதால் Hai-ஐ வெளிப்படுத்தவில்லை.

```
nithya@nithya-laptop:~/Desktop/selenium/Programs$ python Assertions.py
test_register (__main__.Login) ... ok

----------------------------------------------------------------------
Ran 1 test in 16.120s

OK
```

4. assertNotEqual (a,b,[msg]) இது இரண்டு வார்த்தைகளும் சமமாக இல்லையா என சோதிக்கும். அவ்வாறு சமமாக இருந்தால் Hai-ஐ வெளிப்படுத்தும்.

```
self.assertNotEqual("Create New Customer Account",self.a.title,["Hai"])
```

இங்கு இரண்டும் சமமாக உள்ளதால் Hai-ஐ வெளிப்படுத்தியுள்ளது.

```
nithya@nithya-laptop:~/Desktop/selenium/Programs$ python Assertions.py
test_register (__main__.Login) ... FAIL

======================================================================
FAIL: test_register (__main__.Login)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "Assertions.py", line 21, in test_register
    self.assertNotEqual("Create New Customer Account",self.a.title,["Hai"])
AssertionError: ['Hai']

----------------------------------------------------------------------
Ran 1 test in 16.030s

FAILED (failures=1)
```

5. *assertFalse (a, [msg]) விண்ணப்பப் படிவத்தில் 'Sign Up for Newsletter' எனும் checkbox, select செய்யப்பட்டுள்ளது எனும் விவரம் பொய்யா என்பதை சோதிக்கிறது.*

```
self.assertFalse(su.is_selected(),['Hai'])
```

அது பொய் தான் என்பதனால் 'Hai' என்பது வெளிப்படவில்லை.

```
nithya@nithya-laptop:~/Desktop/selenium/Programs$ python Assertions.py
test_register (__main__.Login) ... ok

----------------------------------------------------------------------
Ran 1 test in 15.184s

OK
```

6. assertTrue (a, [msg]) அதே checkbox, select செய்யப்பட்டுள்ளது எனும் விவரம் உண்மையா என்பதை சோதிக்கிறது.

```
self.assertTrue(su.is_selected(),['Hai'])
```

அது பொய்யாக இருக்கவே பின்வருமாறு error-ஐ வெளிப்படுத்தி 'Hai' - ஐயும் வெளிப்படுத்துகிறது.

```
nithya@nithya-laptop:~/Desktop/selenium/Programs$ python Assertions.py
test_register (__main__.Login) ... FAIL

======================================================================
FAIL: test_register (__main__.Login)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "Assertions.py", line 25, in test_register
    self.assertTrue(su.is_selected(),['Hai'])
AssertionError: ['Hai']

----------------------------------------------------------------------
Ran 1 test in 15.413s

FAILED (failures=1)
```

7. assertListEqual (a,b) இது இரண்டு பட்டியல்களை ஒப்பிட்டு அதற்குள் உள்ள மதிப்புகள் அனைத்தும் ஒத்துப் போகிறதா என்பதை சோதிக்கிறது. இதற்கான உதாரணம் இந்தப் புத்தகத்தில் உள்ள வேறுசில *program*-ல் உள்ளது.

3.9 **Dropdown box**- ஐ அணுகுதல்

ஒரு *dropdown box*-க்குள் நாம் எதிர்பார்க்கும் மதிப்புகள் அனைத்தும் இருக்கின்றனவா, *default*-ஆக எந்த மதிப்பு வெளிப்படுகிறது, அவற்றை மாற்றுவது எப்படி என்றெல்லாம் பின்வரும் நிரலில் காணலாம்.






```
#File : 9-dropdown.py

import unittest
from selenium import webdriver
from selenium.webdriver.support.ui import Select

class dropdown (unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        cls.a = webdriver.Firefox()
        cls.a.implicitly_wait(30)
        cls.a.get("http://magento-demo.lexiconn.com/")
        cls.a.maximize_window()

    def test_language_options(self):
            x = ["ENGLISH", "FRENCH", "GERMAN"]
      y = []
      d = Select(self.a.find_element_by_id("select-language"))
      self.assertEqual(3, len(d.options))
      for i in d.options:
          y.append(i.text)
      self.assertListEqual(x, y)
      self.assertEqual("ENGLISH", d.first_selected_option.text)
      d.select_by_visible_text("German")
      self.assertTrue("store=german" in self.a.current_url)
      d = Select(self.a.find_element_by_id("select-language"))
      d.select_by_index(0)

    @classmethod
    def tearDownClass(cls):
      cls.a.quit()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

இங்கு “select-language” dropdownbox -ஆனது d என வைக்கப்பட்டுள்ளது. இந்த பெட்டியில் நாம் எதிர்பார்க்கும் மதிப்புகள் அனைத்தும் x எனும் பெயர் கொண்ட பட்டியலிலும், ஒரு வெற்றுப் பட்டியல் y எனவும் வைக்கப்பட்டுள்ளது.

1. len(d.options) மூலம் d பெட்டிக்குள் உள்ள மதிப்புகளின் எண்ணிக்கை கண்டுபிடிக்கப்பட்டு, அது 3-ஆக உள்ளதா என சோதிக்கப்பட்டுள்ளது. இதன் மூலம் dropdown box-ல் 3 மதிப்புகள் உள்ளது உறுதிசெய்யப்படுகிறது.

2. For loop மூலம் d பெட்டியில் உள்ள மதிப்புகள் ஒவ்வொன்றாக எடுக்கப்பட்டு, அவை y எனும் வெற்றுப் பட்டியலில் இணைக்கப்பட்டுள்ளன. பின்னர் x-ம் y-ம் சமமாக உள்ளதா என்பதை சோதிக்க assertListEqual பயன்பட்டுள்ளது. இதன் மூலம் நாம் எதிர்பார்க்கும் மதிப்புகள் dropdown box-க்குள் உள்ளதா என்பது உறுதிசெய்யப்படுகிறது.

3. பெட்டிக்குள் முதலில் உள்ள மதிப்பு "English" என உள்ளதா என்பது first_selected_option மூலம் சோதிக்கப்பட்டுள்ளது.

4. பின்னர் select_by_visible_text மூலம் அந்த மதிப்பு German என மாற்றப்பட்டுள்ளது. இவ்வாறு மொழியினை மாற்றியவுடன் அந்தப் பக்கத்தில் உள்ள அனைத்தும், ஜெர்மன் மொழிக்கு மாற்றப்பட்டுவிட்டதா என்பதை நமது browser url-ல் சென்று store=german என உள்ளதா என்று பரிசோதிப்பதன் மூலம் உறுதி படுத்தலாம்.

இந்நிலையில் நமது browser url முழுவதுமாக மாற்றப்பட்டு விட்டதால், மீண்டும் “select-language” dropdownbox -ஐ d எனவைப்பதற்கான code கொடுக்கப்பட்டு, அதன் இயல்பான மதிப்பு தேர்ந்தெடுக்கப்படுகிறது.

இப்போது இந்த d பெட்டியின் முக்கிய சில properties / functions ஐப் பின்வருமாறு காணலாம்.

d.options = dropdown பெட்டியில் உள்ள அனைத்து மதிப்புகளையும் எடுத்து ஒரு பட்டியலாக சேமித்துக் கொள்ளும்.

d.all_selected_options = dropdown பெட்டியில் உள்ள மதிப்புகளில் தேர்ந்தெடுக்கப்பட்ட நிலையில் இருக்கும் மதிப்புகளை மட்டும் எடுத்து ஒரு பட்டியலாக சேமித்துக் கொள்ளும்.

d.first_selected_option = dropdown பெட்டியில் உள்ள தேர்ந்தெடுக்கப்பட்ட

நிலையில் இருக்கும் மதிப்புகளில் முதல் மதிப்பினை மட்டும் எடுக்க உதவும்.

d.deselect_all() = ஒரு *multiselect dropdownbox / list* -ல் தேர்ந்தெடுக்கப்பட்ட நிலையில் இருக்கும் மதிப்புகளின் தேர்வுகளை மட்டும் நீக்கி விட உதவும்.

d.deselect_by_ index(xxx) = நாம் கொடுக்கும் *index*-ன் அடிப்படையில் தேர்வுகளை நீக்க உதவும்.

d.deselect_by_ value(xxx) = நாம் கொடுக்கும் மதிப்பின் அடிப்படையில் தேர்வுகளை நீக்க உதவும்.

d.deselect_by_ visible_text(xxx) = நாம் கொடுக்கும் *text*-ஐத் தேடி அதன் மீது உள்ள தேர்வினை நீக்கும்.

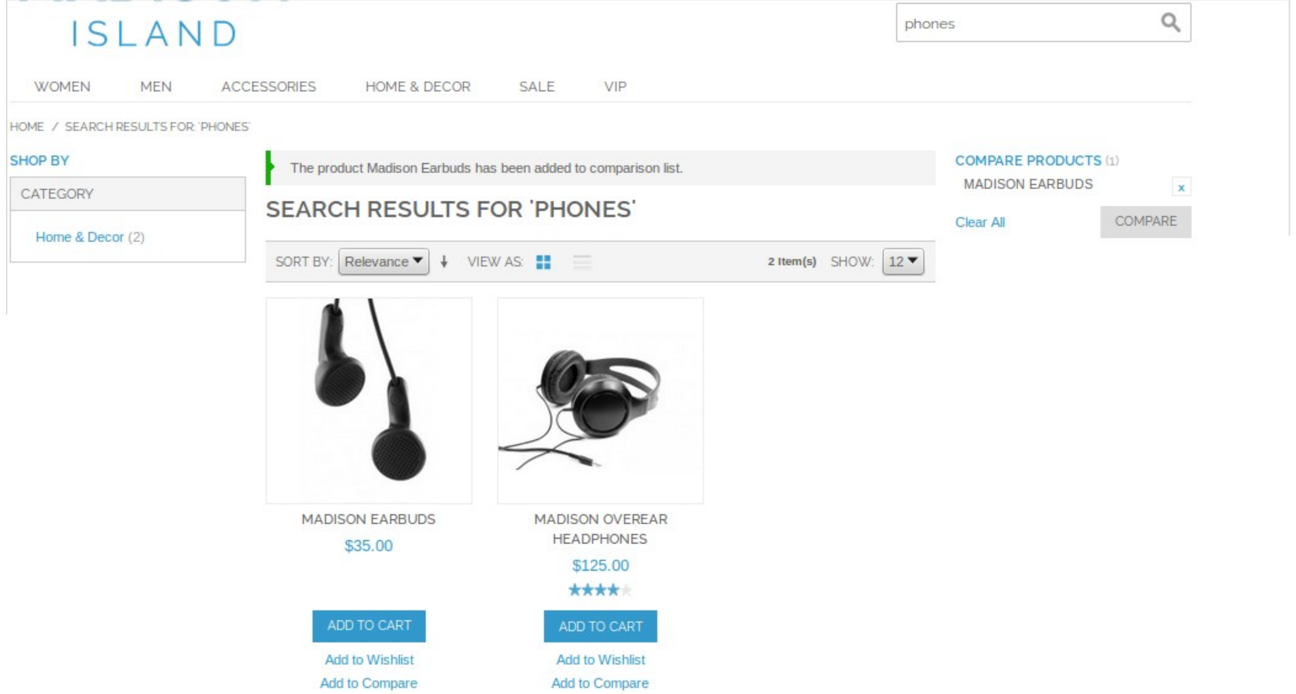
d.select_by_index(xxx) = நாம் கொடுக்கும் *index*-ன் இடத்தில் இருக்கும் மதிப்பினை தேர்வு செய்யும்.

d.select_by_value(xxx) = நாம் கொடுக்கும் மதிப்பின் அடிப்படையில் தேர்வு செய்யும்.

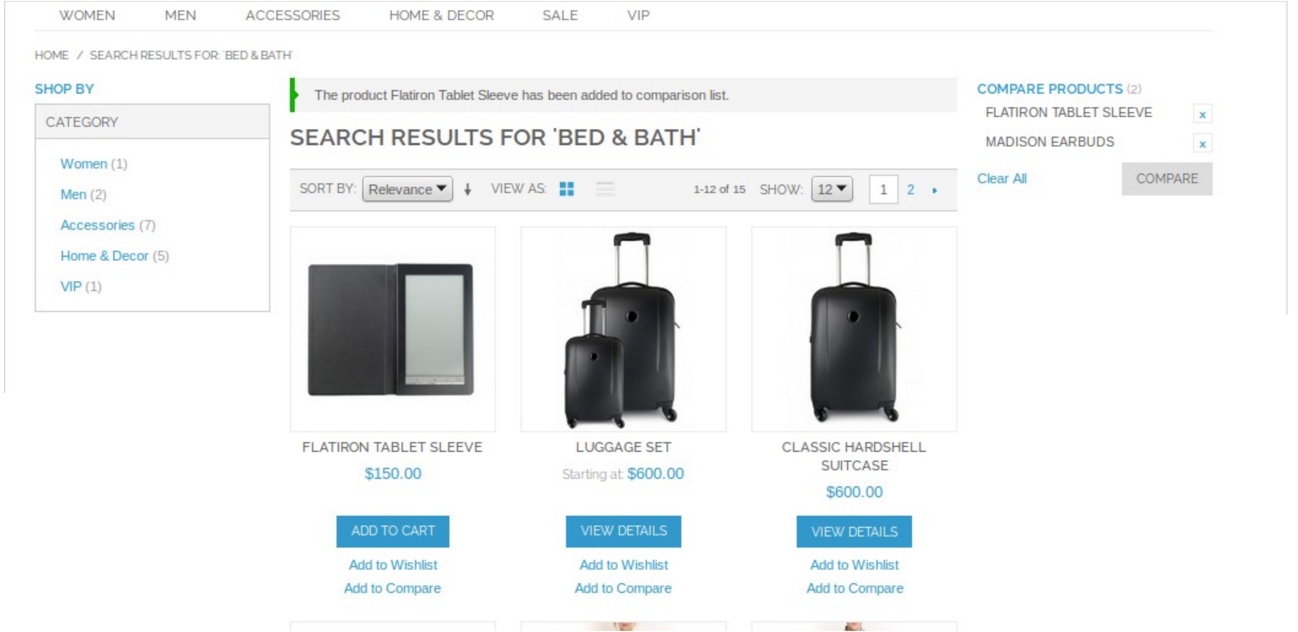
d.select_by_ visible_text(xxx) = நாம் கொடுக்கும் *text*-ஐத் தேடி அதன் அடிப்படையில் தேர்வு செய்யும்.

3.10 **popup**- ஐ கையாளுதல்

Magento தளத்தில் சென்று *phones* எனக் கொடுத்து தேடி, அதன் கீழ் வரும் ஏதேனும் ஒரு பொருளின் அடியில் உள்ள *Add to Compare* இணைப்பின் மீது சொடுக்கவும்.



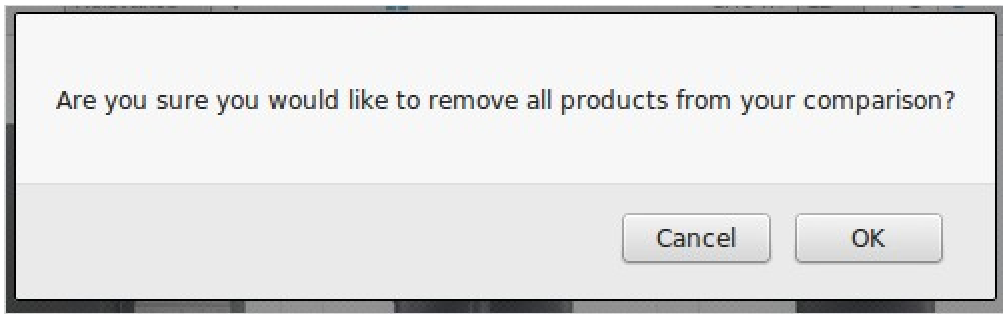
அவ்வாறே *'Bed & Bath'* எனக் கொடுத்து தேடி அதன் கீழ் வரும் ஏதேனும் ஒரு பொருளின் அடியில் உள்ள *Add to Compare* இணைப்பின் மீது சொடுக்கவும்.



இப்போது நாம் தேர்வு செய்துள்ள அந்த இரண்டு பொருள்களும் திரையின் வலப்பக்கத்தில் உள்ள *'Compare Products'* எனும் பகுதியின் கீழ் தெரியும்.



இந்தப் பகுதியில் *Clear All* எனும் இணைப்பின் மீது சொடுக்கினால் பின்வருமாறு ஒரு *Popup* வெளிப்படும்.



இந்த *popup*-ல் உள்ள *ok button*-ஐ எவ்வாறு சொடுக்குவது? இது நமது *browser*-ஐயே *disable* செய்துவிட்டு வெளிப்படுகிறது. இதற்கான *html code* எதுவும் கிடையாது. இது போன்ற ஒரு *popup*-ல் உள்ள *button*-ஐ எவ்வாறு சொடுக்குவது

என்பதற்கான நிரலை பின்வருமாறு காணலாம்.

```
File : 10-compare.py

import unittest
from selenium import webdriver

class compare (unittest.TestCase):
    def setUp(self):
        self.a = webdriver.Firefox()
        self.a.implicitly_wait(30)
        self.a.get("http://magento-demo.lexiconn.com/")
        self.a.maximize_window()

    def test_compare_products(self):

      self.a.find_element_by_xpath("//input[@id='search']").send_keys("phones")
      self.a.find_element_by_xpath("//button[@title='Search']").click()
      self.a.find_element_by_link_text("Add to Compare").click()
      self.a.find_element_by_xpath("//input[@id='search']").clear()
      self.a.find_element_by_xpath("//input[@id='search']").send_keys("Bed &
Bath")
      self.a.find_element_by_xpath("//button[@title='Search']").click()
      self.a.find_element_by_link_text("Add to Compare").click()
      self.a.find_element_by_link_text("Clear All").click()
      i = self.a.switch_to_alert()
      self.assertEqual("Are you sure you would like to remove all products from
your comparison?", i.text)
      i.accept()

    def tearDown(self):
      self.a.quit()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

இங்கு switch_to_alert() எனும் function மூலம் browser-ல் வெளிப்படும் popup ஆனது i எனும் variable-ல் கைப்பற்றப்படுகிறது. பின்னர் popup-ன் முக்கிய சில properties / functions பின்வருமாறு கொடுக்கப்பட்டுள்ளது. இதில் ஒருசில மேற்கண்ட program-ல் பயன்படுத்தப்பட்டுள்ளது.

i.text = popup ல் வெளிப்படும் தகவலை எடுக்க உதவும்.

i.accept() = ok வை சொடுக்கி அதனை ஏற்றுக்கொள்ள உதவும்.

i.dismiss() = cancel ஐ சொடுக்கி அதனை நிராகரிக்க உதவும்.

i.send_keys(xxx) = நம்மிடம் கடவுச் சொல் அல்லது வேறு ஏதாவது ஒரு தகவலைக் கேட்பது போன்று ஒரு popup வெளிப்படின், அந்த மதிப்பினை கொடுக்க உதவும். xxx என்பது நாம் கொடுக்க வேண்டிய மதிப்பு.

3.11 **Hard Coding**

Magento தளத்தில் சென்று "Bed & Bath" எனக் கொடுத்து தேடினால் வெளிப்படும் இணைப்புகளின் எண்ணிக்கை 12-ஆக உள்ளதா மற்றும் "Bags & Luggage" எனக் கொடுத்து தேடும் போது வெளிப்படும் இணைப்புகளின் எண்ணிக்கை 11-ஆக உள்ளதா என்றெல்லாம் சோதிப்பதற்கு ஒரே வகையான code-ஆனது data-வை மட்டும் மாற்றி திரும்பத் திரும்ப பின்வருமாறு கொடுக்கப்பட்டுள்ளது.

```
import unittest
from selenium import webdriver

class search (unittest.TestCase):
    def setUp(self):
        self.a = webdriver.Firefox()
        self.a.implicitly_wait(30)
        self.a.get("http://magento-demo.lexiconn.com/")
        self.a.maximize_window()

    def test_search(self):
        self.a.find_element_by_xpath("//input[@id='search']").send_keys("Bed &
Bath")
        self.a.find_element_by_xpath("//input[@id='search']").submit()
        lis1 = self.a.find_elements_by_xpath("//h2[@class='product-name']/a")
        self.assertEqual(12, len(lis1))

        self.a.find_element_by_xpath("//input[@id='search']").clear()
        self.a.find_element_by_xpath("//input[@id='search']").send_keys("Bags &
Luggage")
        self.a.find_element_by_xpath("//input[@id='search']").submit()
        lis2 = self.a.find_elements_by_xpath("//h2[@class='product-name']/a")
        self.assertEqual(12, len(lis2))

    def tearDown(self):
      self.a.quit()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

இதே போன்று 10 பொருள்களை நான் சோதிக்க விரும்பினால் ஒரே வகையான code-ஐ 10 முறை நான் திரும்ப திரும்ப கொடுக்க வேண்டியிருக்கும். இதுவே hard coding-ஆல் ஏற்படும் பிரச்சனை ஆகும். Hard Coding என்பது நாம் பயன்படுத்தும் data-வை நேரடியாக நிரலுடன் இணைத்து எழுதுவது.

3.12 **Data Driven Testing - Parameters Passing**

மேற்கண்ட பகுதியில் சொன்னது போல், 10 பொருள்களை சோதிப்பதற்கு 10 முறை code-ஐ திரும்ப திரும்ப எழுதாமல், ஒரே ஒரு முறை மட்டும் எழுதலாம். அவ்வாறு எழுதும் பொழுது data-வை நேரடியாக குறிப்பிடாமல் ஒரு variable-ஆக குறிப்பிடலாம். பின்னர் அந்த variable-ன் வழியே ஒவ்வொரு data-ஆக நிரலுக்குள் செலுத்தலாம். இது பின்வருமாறு.

```
import unittest
from selenium import webdriver
from ddt import ddt, data, unpack

@ddt
class search (unittest.TestCase):
    def setUp(self):
        self.a = webdriver.Firefox()
        self.a.implicitly_wait(30)
        self.a.get("http://magento-demo.lexiconn.com/")
        self.a.maximize_window()
    @data (("Bed & Bath",12),("Bags & Luggage",12))
    @unpack
    def test_search(self,i,j):
        self.a.find_element_by_xpath("//input[@id='search']").send_keys(i)
        self.a.find_element_by_xpath("//input[@id='search']").submit()
        lis1 = self.a.find_elements_by_xpath("//h2[@class='product-name']/a")
        self.assertEqual(j, len(lis1))

    def tearDown(self):
      self.a.quit()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

இதில் தேடப்படும் பொருள்களின் பெயரையும், அது வெளிப்படுத்தும் இணைப்புகளின் எண்ணிக்கைக்கான எண்ணையும் நேரடியாக நிரலுக்குள் கொடுக்காமல் முறையே i மற்றும் j எனும் பெயரில் கொடுத்துள்ளோம். இது test_search method-ன் argument-ஆக நிரலுக்குள் செலுத்தப்படுகிறது.

பொருள்களின் பெயர் மற்றும் இணைப்புகளின் எண் ஆகியவற்றை மேற்கண்ட program-ல் கண்டது போல @data-வைத் தொடர்ந்து கொடுக்க வேண்டும். அதனடியில் உள்ள @unpack என்பது dataset-ஐ ஒவ்வொன்றாகப் பிரித்து program-க்குள் செலுத்த உதவுகிறது. இதற்குத் தேவையான ddt modules தொடக்கத்தில் import செய்யப்பட்டிருப்பதைக் காணலாம்.

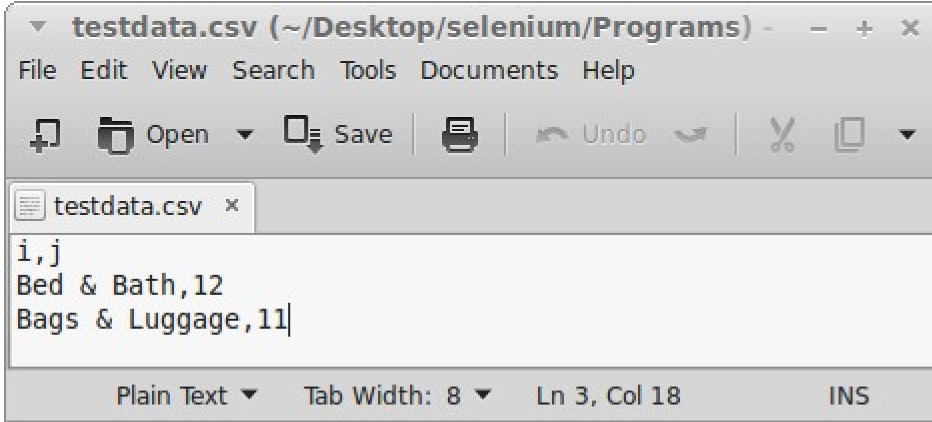
```
sudo pip install ddt
```

என்பது இதற்குத் தேவையான *modules*-ஐ *install* செய்ய உதவும்.

இதுவே *data-driven testing* எனப்படும்.

3.13 Data Driven Testing - Parameters Passing - textfile

மேற்கண்ட அதே *program*-ல் ஒரு சிறிய மாற்றத்தை இந்தப் பகுதியில் காணலாம். *data*-வை எல்லாம் தனியாக ஒரு *textfile*-ல் வைத்து, அதிலிருந்து *program*-க்குள் செலுத்துமாறு செய்வதற்கான *code* பின்வருமாறு அமையும்.



மேற்கண்ட முறையிலேயே *notepad*-ன் முதல் வரியில் *variable*-ஐயும், அடுத்தடுத்த வரிகளில் *data*-வையும் கொடுக்க வேண்டும். கொடுக்கப்படும் *variable* மற்றும் *data* களுக்கு இடையில் *comma(,)* வைத்துக் கொடுக்க வேண்டும்.

```
import csv, unittest
from selenium import webdriver
from ddt import ddt, data, unpack

def get_data(f_name):
    data=[]
    cont = csv.reader(open(f_name, "rb"))
    next(cont, None)
    for i in cont:
        data.append(i)
    return data

@ddt
class search (unittest.TestCase):
    def setUp(self):
        self.a = webdriver.Firefox()
        self.a.implicitly_wait(30)
        self.a.get("http://magento-demo.lexiconn.com/")
        self.a.maximize_window()
```

```
    @data (*get_data("testdata.csv"))



    @unpack
    def test_search(self,i,j):
        self.a.find_element_by_xpath("//input[@id='search']").send_keys(i)
        self.a.find_element_by_xpath("//input[@id='search']").submit()
        lis1 = self.a.find_elements_by_xpath("//h2[@class='product-name']/a")
        self.assertEqual(int(j), len(lis1))

    def tearDown(self):
      self.a.quit()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

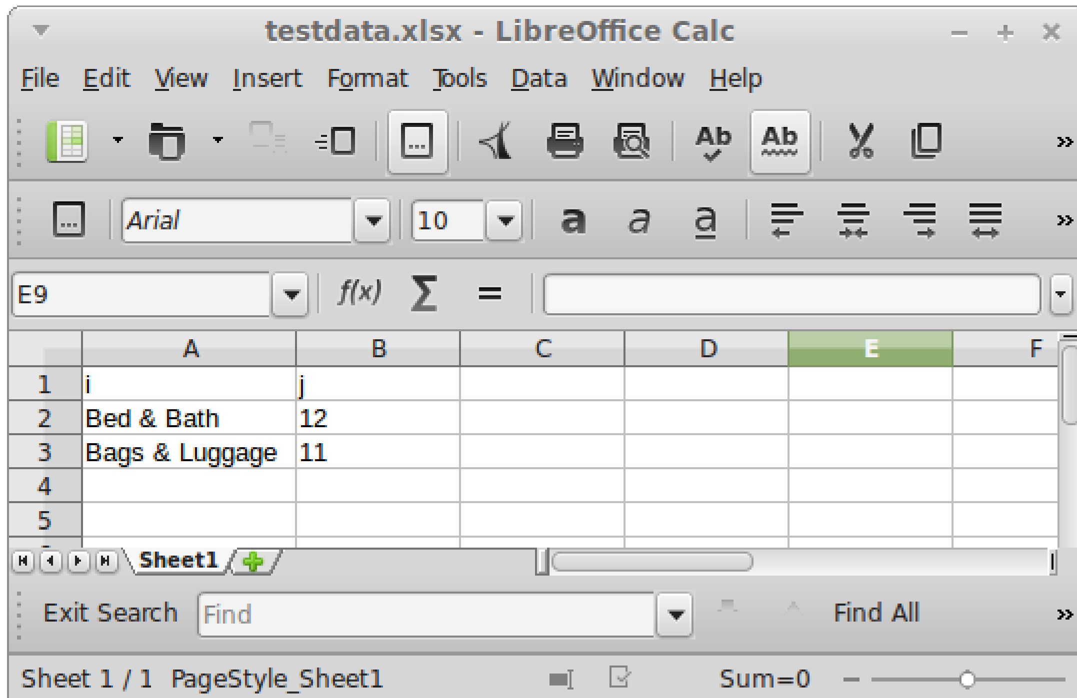
இங்கு test_search எனும் function-க்குள் உள்ள நிரல் எதுவும் மாறவில்லை. மாறாக @data-வைத் தொடர்ந்து, பொருள்களின் பெயர் மற்றும் இணைப்புகளின் எண் ஆகியவை சேமித்து வைக்கப்பட்டுள்ள கோப்பின் பெயர் (testdata.csv) கொடுக்கப்பட்டுள்ளது. அந்த கோப்பின் பெயரை நாம் .csv என்றே சேமிக்க வேண்டும்.

3.14 **Data Driven Testing - Parameters Passing - Excelsheet**

textfile-க்கு பதிலாக excelsheet-ல் இருந்து data-வை எடுப்பதற்கான code பின்வருமாறு அமையும்.

```
sudo pip install xlrd
```

என்பது இதற்குத் தேவையான modules-ஐ install செய்ய உதவும்.



மேற்கண்ட முறையிலேயே Excel-ன் முதல் வரியில் variable-ஐயும், அடுத்தடுத்த வரிகளில் data-வையும் கொடுக்க வேண்டும்.

```
import xlrd, unittest
from selenium import webdriver
from ddt import ddt, data, unpack

def get_data(f_name):
    data=[]
    cont = xlrd.open_workbook(f_name).sheet_by_index(0)
    for i in range(1,cont.nrows)               :
        data.append(list(cont.row_values(i, 0, cont.ncols)))
    return data
```

```
@ddt
class search (unittest.TestCase):
    def setUp(self):
        self.a = webdriver.Firefox()
        self.a.implicitly_wait(30)
        self.a.get("http://magento-demo.lexiconn.com/")
        self.a.maximize_window()
    @data (*get_data("testdata.xlsx"))
    @unpack
    def test_search(self,i,j):
        self.a.find_element_by_xpath("//input[@id='search']").send_keys(i)
        self.a.find_element_by_xpath("//input[@id='search']").submit()
        lis1 = self.a.find_elements_by_xpath("//h2[@class='product-name']/a")
        self.assertEqual(int(j), len(lis1))

    def tearDown(self):
      self.a.quit()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

இங்கும் *test_search* எனும் *function*-க்குள் உள்ள நிரல் எதுவும் மாறவில்லை. மாறாக *@data*-வைத் தொடர்ந்து, பொருள்களின் பெயர் மற்றும் இணைப்புகளின் எண் ஆகியவை சேமித்து வைக்கப்பட்டுள்ள கோப்பின் பெயர் *(testdata.xlsx)* கொடுக்கப்பட்டுள்ளது. அந்த கோப்பின் பெயரை நாம் *.xlsx* என்றே சேமிக்க வேண்டும்.

3.15 **Implicit Wait**

நமது *program-* ல் இருக்கும் ஒவ்வொரு வரியும் புயல் வேகத்தில் *browser-*ல் வேலைகளைச் செய்து கொண்டிருக்கும்போது, சில சமயம் இணைய இணைப்பின் வேகத்தால் அந்த அளவுக்கு ஈடு கொடுக்க முடியாது. உதாரணத்-துக்கு *google-*ல் சென்று *"selenium webdriver"* எனத் தேடி முதல் விடையின் மீது சொடுக்கவும்; பின்னர் *browser-*ல் *back-*ஐயும் *forward-*ஐயும் *refresh-*ஐயும் அடுத்த-டுத்து சொடுக்கவும் என்பதற்கான *code* பின்வருமாறு அமையும்.

```
import unittest
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions

class compare (unittest.TestCase):
    def setUp(self):
        self.a = webdriver.Firefox()
        self.a.implicitly_wait(30)
        self.a.get("http://www.google.com")
        self.a.maximize_window()

    def test_browser_navigation(self):

self.a.find_element_by_xpath("//input[@title='Search']").send_keys("selenium
webdriver")
      self.a.find_element_by_xpath("//input[@title='Search']").submit()
      self.a.find_element_by_link_text("Selenium WebDriver").click()

      self.a.back()
        self.a.implicitly_wait(60)
      self.a.forward()
        self.a.implicitly_wait(60)
      self.a.refresh()
        self.a.implicitly_wait(60)

    def tearDown(self):
      self.a.quit()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

இதனை நாம் self.a.implicitly_wait(60) எனும் வரிகளையெல்லாம் நீக்கிவிட்டு இயக்கிப் பார்த்தால், program சரிவர இயங்காமல் error-ஐ வெளிப்படுத்தும். ஏனென்றால் back மூலம் பின்னோக்கிச் செல்வதற்கு browser கொஞ்ச நேரமாவது எடுத்துக்கொள்ளும். ஆனால் python program அதனைப் புரிந்து கொள்ளாமல் அடுத்தடுத்த கட்டளைகளை இயக்குவதால் வரும் பிரச்சனைதான் இது.

எனவே ஒவ்வொரு கட்டளையையும் இயக்குவதற்கு முன்னர் browser-க்கு கொஞ்ச நேரமாவது கொடு எனச் சொல்வதற்கு implicitly_wait(60) என்பது பயன்படுகிறது. ஒவ்வொரு கட்டளைக்கும் முன்னர் implicitly_wait()-ஐப் பயன்படுத்துவதன் மூலம் இணையத்தின் வேகத்திற்கு ஏற்றவாறு program-ன் வேகத்தை நாம் குறைக்க முடியும்.

3.16 **Explicit Wait**

Explicit Wait என்பது *browser*-ல் ஒரு குறிப்பிட்ட செயல் நிகழும் வரை *program*-ஐ காத்திருக்க வைக்கும். அதாவது ஒரு குறிப்பிட்ட *element, load*-ஆகும் வரை அல்லது அது *enable*-ஆகும் வரை என்பது போன்ற *conditions*-ன் அடிப்படையில், நமது *program*-ஐ காத்திருக்க வைக்கும். இதற்காக *WebDriverWait* மற்றும் *expected_conditions* போன்ற *classes* பயன்படும். இதைப் பயன்படுத்தி உருவாக்கப்பட்ட *code* பின்வருமாறு.

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions
import unittest

class kanchipuram (unittest.TestCase):
    def setUp(self):
        self.a = webdriver.Firefox()
        self.a.get("http://magento-demo.lexiconn.com/")
        self.a.maximize_window()

    def test_account_link(self):
        WebDriverWait(self.a, 10).until(lambda s: s.find_element_by_id("select-
language").get_attribute("length") == "3")
        b = WebDriverWait(self.a,
10).until(expected_conditions.visibility_of_element_located((By.LINK_TEXT,
"ACCOUNT")))
        b.click()

    def tearDown(self):
        self.a.quit()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

இங்கு *select_language* எனும் பெட்டி *load* ஆகும் வரையும், *'ACCOUNT'* எனும் இணைப்பு தெரியும் வரையும் காத்திருக்குமாறு சொல்லப்பட்டுள்ளது. இதனை நாம் *run* செய்யும்போது, நமது இணைய இணைப்பு மிகவும் மெதுவாக இருந்தால் ஒழிய நமக்கு ஒரு வித்தியாசமும் தெரியாது. இருந்தாலும் இப்படியெல்லாம் ஒரு விஷயம் உள்ளது என்பதை சொல்லியுள்ளேன்.

Expected_conditions class-ல் பயன்படுத்தப்படும் ஒருசில முக்கியமான *condition* வகைகளைப் பின்வரும் பட்டியலில் காணலாம்.

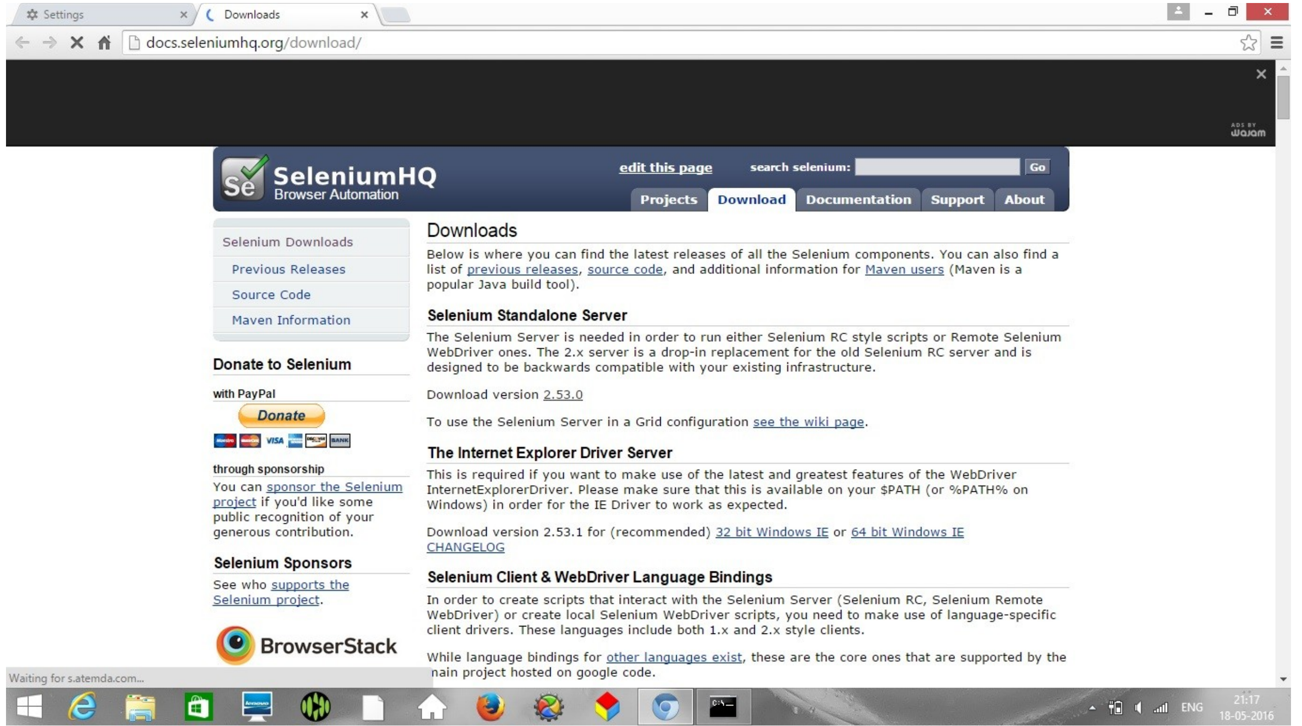
Expected Condition	உதாரணம்
element_to_be_clickable()	WebDriverWait(self.a,10).until(expected_conditions.element_to_be_clickable((By.NAME,"is_subscribed")))
element_to_be_selected()	subscription = self.driver.find_element_by_name("is_subscribed")WebDriverWait(self.driver, 10).until(expected_ conditions.element_to_be_selected(subscription))
invisibility_of_element_located()	WebDriverWait(self.a, 10).until(expected_conditions.invisibility_of_element_located((By.ID,"loading_banner")))
presence_of_all_elements_located()	WebDriverWait(self.a, 10).until(expected_conditions.presence_of_all_elements_located((By.CLASS_NAME,"input-text")))
presence_of_element_located()	WebDriverWait(self.a, 10).until(expected_conditions.presence_of_element_located((By.ID,"search")))
text_to_be_present_in_element()	WebDriverWait(self.a,10).until(expected_conditions.text_to_be_present_in_element((By.ID,"selectlanguage"),"English"))
title_contains()	WebDriverWait(self.a, 10).until(expected_conditions.title_contains("Create New Customer Account"))
title_is()	WebDriverWait(self.a, 10).until(expected_conditions.title_is("Create New Customer Account - Magento Commerce Demo Store"))
visibility_of()	first_name = self.a.find_element_by_id("firstname")WebDriverWait(self.driver, 10).until(expected_conditions.visibility_of(first_name))
visibility_of_element_located()	WebDriverWait(self.a, 10).until(expected_conditions.visibility_of_element_located((By.ID,"firstname")))

4 Selenium Standalone Server

‘Selenium Standalone server’ என்பது நமது கணினியில் மட்டுமல்லாமல் மற்ற கணினிகளுடன் தொடர்பு கொண்டு, அதிலும் சோதனைகளை நிகழ்த்தப் பயன்படும் ஒரு கருவி என்பதை நாம் முன்னரே கண்டோம். நான் இந்தப் புத்தகத்தில் உள்ள அனைத்து சோதனைகளையும் எனது லினக்ஸ் கணினியில் உள்ள *firefox*-ல் செய்து காட்டினேன். இப்போது அதில் ஏதாவது ஒரு சோதனையை எனது கணவரின் *windows*-கணினியில் உள்ள *IE*-ல் எவ்வாறு நிகழ்த்துவது என்று செய்து காட்டப் போகிறேன். அதற்கு முதலில் அந்த *windows*-கணினியில் சென்று *Selenium Standalone server*-ஐ நிறுவ வேண்டும்.

4.1 Standalone Server-ஐ நிறுவுதல் (In Windows)

http://docs.seleniumhq.org/download/ எனும் முகவரியில் சென்று *standalone server 2.53.0* மற்றும் *The IE driver server 2.53.1* ஆகியவற்றை பதிவிறக்கம் செய்யவும்.



பின்னர் அவற்றை *unzip* செய்து அதிலுள்ள கோப்புகளை இயக்கி C:\ -க்குள் சேமித்துக் கொள்ளவும். பின்னர் *command prompt*-ல் *C :\>* க்குள் சென்று பின்வரும் *command*-ஐ இயக்கவும்.

```
java -Dwebdriver.ie.driver="C:\IEDriverServer.exe"  -jar selenium-server-standalone-2.53.0.jar
```

இது *"Selenium Server is up and running"* என்பதனை *command prompt*-ல் வெளிப்படுத்தும். அவ்வளவுதான்! *Selenium Standalone Server*-ஆனது இப்போது *Windows* கணினியில் இயங்கிக் கொண்டிருக்கிறது. இப்போது நமது லினக்ஸ் கணினியில் சென்று *program*-ஐ இயக்கினால் அது *windows* கணினியில் இயங்குமாறு செய்ய என்ன செய்ய வேண்டும் என்பதை கீழ் வரும் பகுதியில் காணலாம்.

4.2 **Standalone Server**-ஐ தொடர்பு கொண்டு சோதனைகளை இயக்குதல் (**From Linux**)

பகுதி 3.2-ல் கொடுக்கப்பட்டுள்ள அதே program- ஐ windows- ல் இயங்குமாறு செய்ய அதன் setUp class-ஐ மட்டும் பின்வருமாறு மாற்றி அமைக்க வேண்டும்.

```
import unittest
from selenium import webdriver

class SearchTests (unittest.TestCase):
    def setUp(self):
      desired_caps = {}
      desired_caps['platform'] = 'WINDOWS'
      desired_caps['browserName'] = 'internet explorer'
      self.a = webdriver.Remote('http://192.168.1.5:4444/wd/hub',desired_caps)
      self.a.ignoreZoomSetting=True
        self.a.get("http://magento-demo.lexiconn.com/")
        self.a.maximize_window()


    def test_search(self):
      self.a.find_element_by_xpath("//input[@id='search']").send_keys("Bed &
Bath")
      self.a.find_element_by_xpath("//input[@id='search']").submit()
      lis = self.a.find_elements_by_xpath("//h2[@class='product-name']/a")
      self.assertEqual(12, len(lis))

    def tearDown(self):
      self.a.close()

if __name__ == '__main__':
      unittest.main(verbosity=2)
```

இதில் Windows கணினியின் IP address கொடுக்கப்பட்டுள்ளது. எனவே இதனை நீங்கள் சாதாரணமாக எப்போதும்போல் உங்கள் கணினியில் இருந்து இயக்குங்கள். அது தானாகவே windows-கணினியில் சென்று இயங்க ஆரம்பித்துவிடும்.

5 Selenium Grid

'Selenium Grid' என்பது ஒரே நேரத்தில் பல்வேறு கணினிகளுடன் தொடர்பு கொண்டு, நமது சோதனைகளை பல்வேறு வகையான *browser*-களில் நிகழ்த்த உதவும் என்பதை நாம் முன்னரே கண்டோம்.

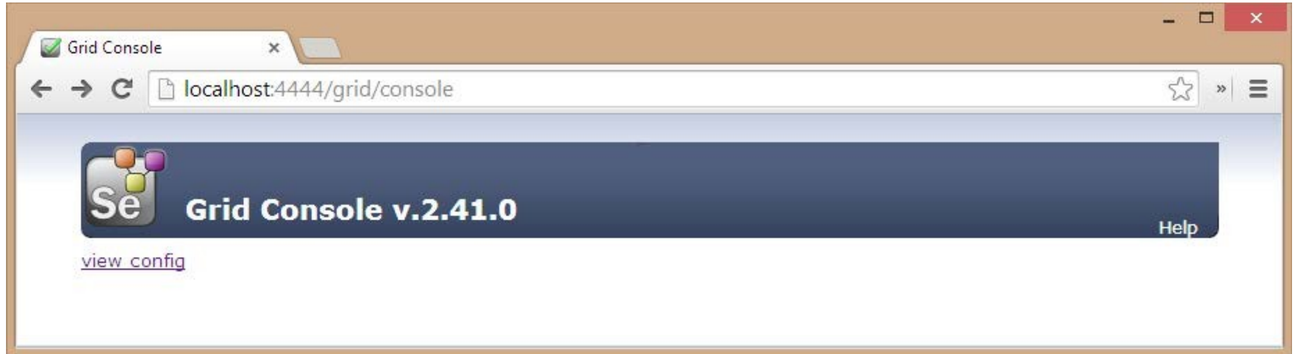
5.1 Selenium Grid-ஐ நிறுவுதல்

Grid-ஐ இயக்க தனியாக மென்பொருள் எதுவும் தேவை இல்லை. *Standalone server*-ஏ போதும். பின்வரும் *command*-ஐ இயக்கவும்.

```
java -jar selenium-server-standalone-2.25.0.jar -port 4444 -role hub
```

இதில் *-port 4444 -role hub* என்பதை கவனிக்க.

Browser-ல் இதன் வெளிப்பாடு .



5.1.2 Node -களை இணைத்தல்

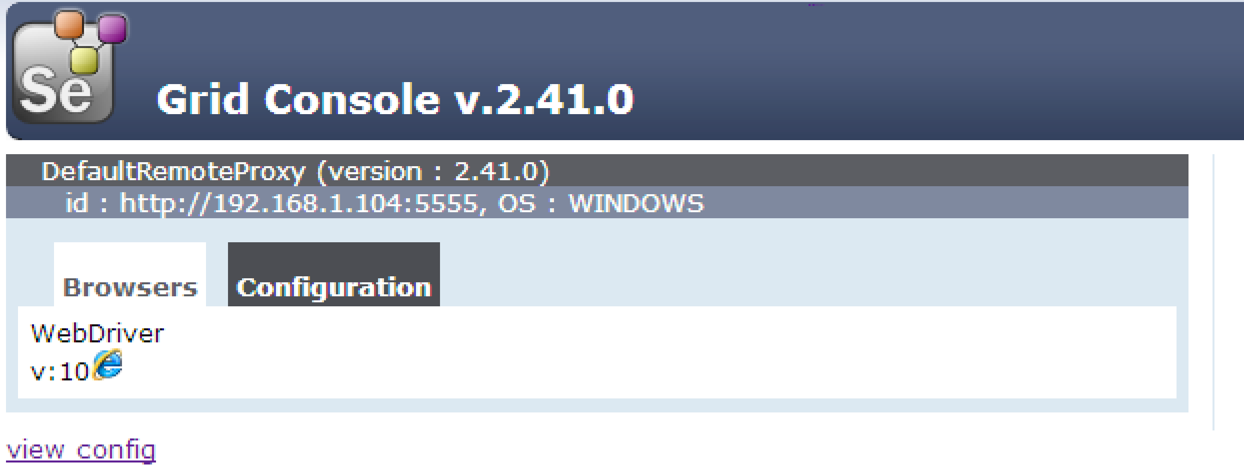
ஒரு *Grid*-உடன் பல்வேறு இயக்குதளங்களில் பல்வேறு உலாவிகளை*(Nodes)* இணைக்க முடியும். அவற்றின் செயல்முறையை பார்ப்போம்.

5.1.3 IE node-ஐ இணைத்தல்

Grid-உடன் *Windows* கணினியின் *IE* உலாவியை இணைப்போம்.

```
java -Dwebdriver.ie.driver="C:\SeDrivers\IEDriverServer.exe" -jar
selenium-server-standalone-2.41.0.jar -role webdriver -browser
```

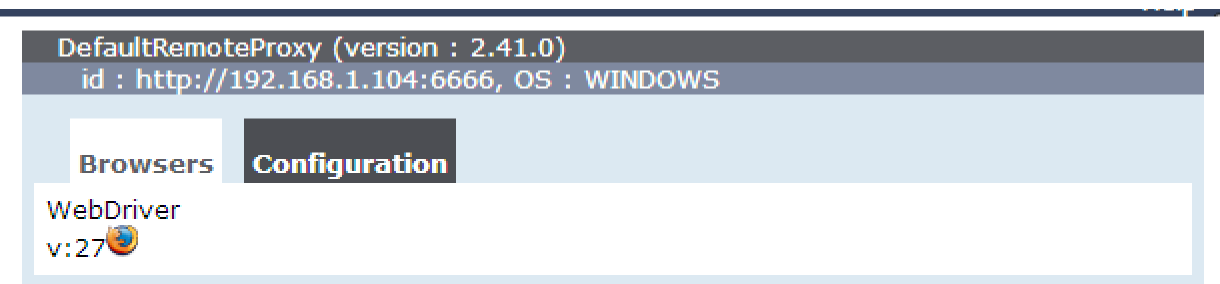
```
"browserName=internet explorer,version=10,maxinstance=1,platform=WINDOWS"
-hubHost 192.168.1.103 -port 5555
```



5.1.4 Firefox node-ஐ இணைத்தல்

Grid-உடன் Windows கணினியின் Firefox உலாவியை இணைப்போம்.

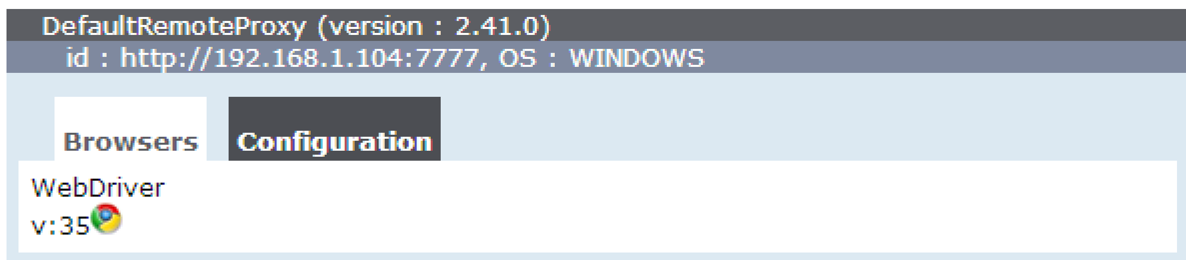
```
java -jar selenium-server-standalone-2.41.0.jar -role webdriver -browser
"browserName=firefox,version=27,maxinstance=2,platform=WINDOWS" -hubHost
localhost -port 6666
```



5.1.5 Chrome node-ஐ இணைத்தல்

Grid-உடன் Windows கணினியின் Chrome உலாவியை இணைப்போம்.

```
java -Dwebdriver.chrome.driver="C:\SeDrivers\chromedriver.exe" -jar
selenium-server-standalone-2.41.0.jar -role webdriver -browser "browserN
ame=chrome,version=35,maxinstance=2,platform=WINDOWS" -hubHost localhost
-port 7777
```



5.1.6 Mac OS X உடன் safari-ஐ இணைத்தல்

Grid-உடன் Mac OS X கணினியின் safari உலாவியை இணைப்போம்.

```
java -jar selenium-server-standalone-2.41.0.jar -role webdriver -browser
"browserName=safari,version=7,maxinstance=1,platform=MAC" -hubHost
192.168.1.104 -port 8888
```



Grid Console v.2.41.0 Help

DefaultRemoteProxy (version : 2.41.0)
id : http://192.168.1.104:5555, OS : WINDOWS

Browsers Configuration

WebDriver
v:10

DefaultRemoteProxy (version : 2.41.0)
id : http://192.168.1.104:7777, OS : WINDOWS

Browsers Configuration

WebDriver
v:35

DefaultRemoteProxy (version : 2.41.0)
id : http://192.168.1.104:6666, OS : WINDOWS

Browsers Configuration

WebDriver
v:27

DefaultRemoteProxy (version : 2.41.0)
id : http://192.168.1.100:8888, OS : MAC

Browsers Configuration

WebDriver
v:7

view config

இதே போல் பல்வேறு கணினிகள் பல்வேறு உலாவிகளை இணைத்து ஒரே நேரத்தில் பலநூறு சோதனைகளை இயக்க முடியும்.

5.2 Grid-ஐ தொடர்பு கொண்டு சோதனைகளை இயக்குதல் (From Linux)

இயக்குதளம் , உலாவியின் பெயர் தந்து அது சார்ந்த கணினிகளில் சோதனைகளை இயக்கும் வழியைப் பார்ப்போம்.

பகுதி 3.2-ல் கொடுக்கப்பட்டுள்ள அதே program- ஐ windows- ல் இயங்குமாறு செய்ய அதன் setUp class-ஐ மட்டும் பின்வருமாறு மாற்றி அமைக்க வேண்டும்.

```
#File : 5.2-grid-assersions.py

import sys
import unittest
from selenium import webdriver

class SearchTests (unittest.TestCase):

    PLATFORM = "WINDOWS'
    BROWSER = 'firefox'


    def setUp(self):

        desired_caps = {}
        desired_caps['platform'] = self.PLATFORM
        desired_caps['browserName'] = self.BROWSER

        self.a =
webdriver.Remote('http://192.168.1.104:4444/wd/hub',desired_caps)
        self.a.get("http://magento-demo.lexiconn.com/")
        self.a.implicitly_wait(30)
        self.a.maximize_window()

    def test_search(self):
        self.a.find_element_by_xpath("//input[@id='search']").send_keys("Bed &
Bath")
        self.a.find_element_by_xpath("//input[@id='search']").submit()
        lis = self.a.find_elements_by_xpath("//h2[@class='product-name']/a")
        self.assertEqual(12, len(lis))

    def tearDown(self):
        self.a.close()

if  __name__ == '__main__':
        if len(sys.argv) > 1:
            SearchTests.BROWSER = sys.argv.pop()
            SearchTests.PLATFORM = sys.argv.pop()
        unittest.main(verbosity=2)
```

பின்வருமாறு வெவ்வேறு இயக்குதளம், உலாவிகளில் இதே சோதனையை இயக்கலாம்.

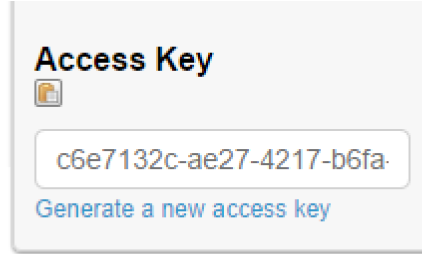
```
python grid_test.py WINDOWS chrome

python grid_test.py MAC safari
```

5.3 இணையம் வழியே சோதனைகளை இயக்குதல்

மேற்கண்ட Grid-ஐ உருவாக்க நமக்கு பல்வேறு கணினிகள் தேவை. நம்மிடம் அவ்வளவு வளங்கள் இல்லாத போது இணைய வழியே கிடைக்கும் சேவையை நாம் பயன்படுத்திக் கொள்ளலாம். இதற்குத் தனிக் கட்டணம் உண்டு. இலவச சேவையாகவும் கிடைக்கிறது.

https://saucelabs.com/ என்ற தளம் இதற்கு உதவுகிறது. அதில் பதிவு செய்து கொள்க. ஒரு Access Key கிடைக்கும்.



உங்களது username, access key - யை நிரலில் சேர்க்க வேண்டும்.

```
#File : 5.3-cloud-assersions.py

import sys
import unittest
from selenium import webdriver

class SearchTests (unittest.TestCase):

    PLATFORM = "WINDOWS'
    BROWSER = 'firefox'
    SAUCE_USERNAME = 'nithya'
    SUACE_KEY = 'c6e7132c-ae27-4217-b6fa-3cf7df0a7281'

    def setUp(self):

        desired_caps = {}
        desired_caps['platform'] = self.PLATFORM
        desired_caps['browserName'] = self.BROWSER

        sauce_string = self.SAUCE_USERNAME + ':' + self.SUACE_KEY
        self.driver = webdriver.Remote('http://' + sauce_string +
'@ondemand.saucelabs.com:80/wd/hub', desired_caps)

        self.a =
webdriver.Remote('http://192.168.1.104:4444/wd/hub',desired_caps)
        self.a.get("http://magento-demo.lexiconn.com/")
        self.a.implicitly_wait(30)
        self.a.maximize_window()
```

```
    def test_search(self):
        self.a.find_element_by_xpath("//input[@id='search']").send_keys("Bed &
Bath")
        self.a.find_element_by_xpath("//input[@id='search']").submit()
        lis = self.a.find_elements_by_xpath("//h2[@class='product-name']/a")
        self.assertEqual(12, len(lis))

    def tearDown(self):
        self.a.close()

if __name__ == '__main__':
        if len(sys.argv) > 1:
            SearchTests.BROWSER = sys.argv.pop()
            SearchTests.PLATFORM = sys.argv.pop()
        unittest.main(verbosity=2)
```

இவ்வாறு நிரலை இயக்கலாம்.

```
python sauce_test.py "OS X 10.9" "Safari"
```

SAS Lab dashboard-ல் தற்போது இயங்கும் நிரல்களைக் காணலாம்.

	Session	Environment	Tags	Build	Results	End	Run Time
	unnamed job	10.9 7			Running View Fullscreen | Spy		

மேலும் selenium logs, outputs, screenshots, videos ஆகியவற்றையும் காணலாம்.

unnamed job
complete

10.9 7 Not on Sauce Connect

by upgundecha
Start Time: 10-12-2014 23:15:01 | Duration: 24s

Commands | Screencast | Selenium Log

POST /session 1s (+3.78s)
desiredCapabilities: {"browserName":"safari","safari.options":{"port":1034}}

=> {"takesScreenshot":true,"cssSelectorsEnabled":true,"ja...

POST url 7s (+2.03s)
url: "http://demo.magentocommerce.com/"

=> ""

POST timeouts/implicit_wait 10s (+0.02s)
ms: 30000

SCR

python Window
demo.magentocommerce.com
Magento
YOUR LANGUAGE: English
MADISON
ISLAND
WOMEN MEN ACCESSORIES HOME & DECOR SAL

6 முடிவுரை

இந்த நூலில் *Selenium* - ன் அடிப்படைகளை மட்டுமே பார்த்துள்ளோம்.

இன்னும் இந்த நூலில் எழுதப் படாதவை பல. அவற்றை வாசகர்கள் இணையத்தில் தேடி, அறிந்து கொள்ள இந்த நூல் ஆர்வத்தைத் தூண்டும் என நம்புகிறேன்.

பின்வரும் இணைப்புகள் மிகவும் பயனுள்ளதாக இருக்கும்.

http://www.guru99.com/selenium-tutorial.html

http://www.tutorialspoint.com/selenium/

https://www.seleniumeasy.com/selenium-tutorials

7. ஆசிரியர் பற்றி

து. நித்யா

கணிணி நுட்பங்களை தமிழில் எழுதி வருகிறேன். Cognizant Technologies Solutions நிறுவனத்தில், Datawarehouse Testing துறையில் பணிபுரிகிறேன்.

"தேமதுரத் தமிழோசை உலகெல்லாம் பரவும் வகை செய்தல் வேண்டும்"
"பிற நாட்டு நல்லறிஞர் சாத்திரங்கள் தமிழ் மொழியிற் பெயர்த்தல் வேண்டும்"

என்ற பாரதியின் விருப்பங்களை நிறைவேற்றுவதில், என் பங்களிப்பும் உள்ளது என்பதே, மிகவும் மகிழ்ச்சி.

இப்போது இங்கிலாந்தில் பணிக்காக உள்ளேன்.

மின்னஞ்சல் - nithyadurai87@gmai.com
வலைப்பதிவு - http://nithyashrinivasan.wordpress.com

8 ஆசிரியரின் பிற மின்னூல்கள்

எளிய தமிழில்



பாகம் - 1

து. நித்யா

கணியம் வெளியீடு

எளிய தமிழில்



பாகம் - 2

து. நித்யா

கணியம் வெளியீடு

http://kaniyam.com



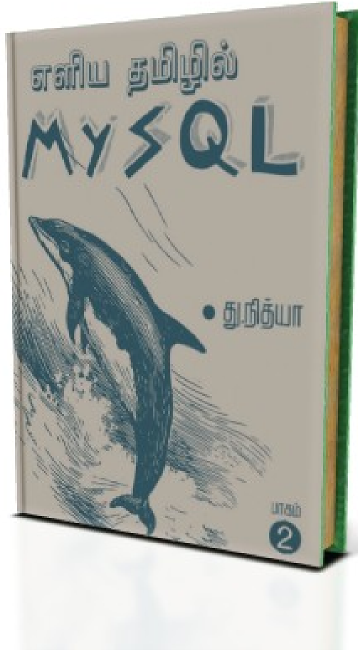
எளிய தமிழில்



து. நித்யா

கணியம் வெளியீடு







http://freetamilebooks.com/ebooks/learn-mysql-in-tamil/

http://freetamilebooks.com/ebooks/learn-mysql-in-tamil-part-2/

http://freetamilebooks.com/ebooks/learn-gnulinux-in-tamil-part1/

http://freetamilebooks.com/ebooks/learn-gnulinux-in-tamil-part2/

http://freetamilebooks.com/ebooks/html-in-tamil/

http://freetamilebooks.com/ebooks/learn-css-in-tamil/

9 கணியம் பற்றி

இலக்குகள்

- கட்டற்ற கணிநுட்பத்தின் எளிய விஷயங்கள் தொடங்கி அதிநுட்பமான அம்சங்கள் வரை அறிந்திட விழையும் எவருக்கும் தேவையான தகவல்களை தொடர்ச்சியாகத் தரும் தளமாய் உருபெறுவது.
- உரை, ஒலி, ஒளி என பல்லூடக வகைகளிலும் விவரங்களை தருவது.
- இத்துறையின் நிகழ்வுகளை எடுத்துரைப்பது.
- எவரும் பங்களிக்க ஏதுவாய் யாவருக்குமான நெறியில் விவரங்களை வழங்குவது.
- அச்சு வடிவிலும், புத்தகங்களாகவும், வட்டுக்களாகவும் விவரங்களை வெளியிடுவது.

பங்களிக்க

- விருப்பமுள்ள எவரும் பங்களிக்கலாம்.
- கட்டற்ற கணிநுட்பம் சார்ந்த விஷயமாக இருத்தல் வேண்டும்.
- பங்களிக்கத் தொடங்கும் முன்னர் கணியத்திற்கு உங்களுடைய பதிப்புரிமத்தை அளிக்க எதிர்பார்க்கப்படுகிறீர்கள்.
- editor@kaniyam.com முகவரிக்கு கீழ்க்கண்ட விவரங்களடங்கிய மடலொன்றை உறுதிமொழியாய் அளித்துவிட்டு யாரும் பங்களிக்கத் தொடங்கலாம்.
    - **மடலின் பொருள்:** பதிப்புரிமம் அளிப்பு
    - **மடல் உள்ளடக்கம்**
        - என்னால் கணியத்திற்காக அனுப்பப்படும் படைப்புகள் அனைத்தும் கணியத்திற்காக முதன்முதலாய் படைக்கப்பட்டதாக உறுதியளிக்கிறேன்.
        - இதன்பொருட்டு எனக்கிருக்கக்கூடிய பதிப்புரிமத்தினை கணியத்திற்கு வழங்குகிறேன்.
        - உங்களுடைய முழுப்பெயர், தேதி.
- தாங்கள் பங்களிக்க விரும்பும் ஒரு பகுதியில் வேறொருவர் ஏற்கனவே பங்களித்து வருகிறார் எனின் அவருடன் இணைந்து பணியாற்ற முனையவும்.
- கட்டுரைகள் மொழிபெயர்ப்புகளாகவும், விஷயமறிந்த ஒருவர் சொல்லக் கேட்டு கற்று இயற்றப்பட்டவையாகவும் இருக்கலாம்.
- படைப்புகள் தொடர்களாகவும் இருக்கலாம்.
- தொழில் நுட்பம், கொள்கை விளக்கம், பிரச்சாரம், கதை, கேலிச்சித்திரம், நையாண்டி எனப் பலசுவைகளிலும் இத்துறைக்கு பொருந்தும்படியான

ஆக்கங்களாக இருக்கலாம்.

- தங்களுக்கு இயல்பான எந்தவொரு நடையிலும் எழுதலாம்.
- தங்களது படைப்புகளை எளியதொரு உரை ஆவணமாக editor@kaniyam.com முகவரிக்குஅனுப்பிவைக்கவும்.
- தள பராமரிப்பு, ஆதரவளித்தல் உள்ளிட்ட ஏனைய விதங்களிலும் பங்களிக்கலாம்.
- ஐயங்களிருப்பின் editor@kaniyam.com மடலியற்றவும்.

விண்ணப்பங்கள்

- கணித் தொழில்நுட்பத்தை அறிய விழையும் மக்களுக்காக மேற்கொள்ளப்படும் முயற்சியாகும் இது.
- இதில் பங்களிக்க தாங்கள் அதிநுட்ப ஆற்றல் வாய்ந்தவராக இருக்க வேண்டும் என்ற கட்டாயமில்லை.
- தங்களுக்கு தெரிந்த விஷயத்தை இயன்ற எளிய முறையில் எடுத்துரைக்க ஆர்வம் இருந்தால் போதும்.
- இதன் வளர்ச்சி நம் ஒவ்வொருவரின் கையிலுமே உள்ளது.
- குறைகளிலிருப்பின் முறையாக தெரியப்படுத்தி முன்னேற்றத்திற்கு வழி வகுக்கவும்.

வெளியீட்டு விவரம்

ஆசிரியர்: த. சீனிவாசன் – editor@kaniyam.com +91 98417 95468

கட்டுரைகளில் வெளிப்படுத்தப்படும் கருத்துக்கள் கட்டுரையாசிரியருக்கே உரியன.

10 நன்கொடை

Creative Commons உரிமையில், யாவரும் இலவசமாகப் பகிரும் வகையில் தமது நூல்களை வெளியிடும் எழுத்தாளரை உங்கள் நன்கொடைகள் ஊக்குவிக்கும்.

வங்கி விவரங்கள்.

Name - Nithya Duraisamy
ICICI - 006101540799
Branch - Mcity branch, chengalpattu.
IFSC code - ICIC0000061